VŠB – Technická univerzita Ostrava
Fakulta Elektroniky a informatiky

Diplomová práce

2011 Jan Smuda

VŠB – Technická univerzita Ostrava
Fakulta Elektroniky a informatiky
Katedra informatiky

Aktualizační server
Updated Server

2011 Jan Smuda

„Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

Datum: ……. Podpis: …….

Souhlasím se zveřejněním této diplomové práce dle požadavku čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava.

Datum: ……. Podpis: …….

Touto cestou, bych rád poděkoval vedoucí diplomové práce, paní RNDr. Daniele Szturcové, PhD. za rady a pomoc při psaní tohoto textu.

Abstrakt

Tato diplomová práce se zabývá návrhem a implementací programového vybavení, které má sloužit jako podpora k již existující aplikaci Agrokrom 6.0. Tento text lze logicky rozdělit na dvě části. První se zabývá licenčními a distribučními politikami na obecné úrovni a dále konkrétními požadavky na licencování a distribuci Agrokromu. Druhá část se zabývá samotnými požadavky, analýzou a návrhem P-Serveru.

Klíčová slova:

Agrokrom 6.0, Aktualizační server, Licence, Aktualizace, Distribuce, RUP, Informační systém, UML, UP

Abstract

This thesis describes the design and implementation of software to serve as support to existing applications Agrokrom 6.0. This text is logically divided into two parts. The first deals with the licensing and distribution policies on a general level and specific requirements for licensing and distribution of Agrokrom. The second part deals with the very requirements analysis and design of the P-Server.

Keywords:

Agrokrom 6.0, Updated server, License, Upgrade, Distribution, RUP, Information system, UML, UP

Seznam použitých symbolů a zkratek

EULA	End user license agreement – Licenční smlouva s koncovým uživatelem
RUP	Rational unified process
UP	Unified process
UML	Unified modeling language
Vukrom	Výzkumný ústav Kroměříž
SAD	Software architecture document
WCF	Windows communication foundation
IE	Internet Explorer
IIS	Internet information service - Internetová informační služba
OSS	Open Source Software – Software s otevřeným zdrojovým kódem
GPL	General Public Licence – Všeobecná veřejná licence
LGPL	Lesser general public licence – Menší (slabší) všeobecná veřejná licence
UI	User interface – Uživatelské rozhraní
CRUD	Create, Update, Delete

Obsah

1 Úvod

Cílem tohoto projektu je analýza a návrh aktualizačního serveru (dále jen P-Server) a rozšíření funkcí stávající aplikace Agrokrom 6.0 tak, aby byla schopna využít služby P-Serveru. Toto musí být založeno na návrhu licenčních a distribučních politik pro Agrokrom 6.0, který bude taktéž potřeba zpracovat.

Základním návrhem licenční politiky se již zabývali tvůrci Agrokromu (viz dokumentace k projektu Agrokrom 6.0). Je potřeba se držet základních principů tohoto návrhu, ale musí být dodělán tak, aby odpovídal nynějším požadavkům. Od tohoto návrhu se bude odvíjet i značná část funkcí, které budou potřebné pro zajištění objednávání, distribuce licencí a aktualizací a tedy ovlivní návrh P-Serveru.

P-Server bude služba poskytovaná v rámci internetu jako podpora zákazníků. Primárně bude sloužit pro poskytování aktualizací pro komponenty Agrokromu, na které bude mít uživatel licenci, pro doobjednání dalších částí systému, pro správu licencí a uživatelských účtů zákazníků. Správcům P-Serveru bude potřeba umožnit publikovat aktualizační balíčky (patch), přidávat nové moduly aplikace a dále spravovat účty a licence zákazníků. Administrační rozhraní aplikace P-Server musí být dostupné pouze z lokální sítě výzkumného ústavu.

Jak již bylo zmíněno, část systému bude dostupná přes internet a uživatelé aplikace Agrokrom se k ní budou připojovat prostřednictvím aplikačního serveru, který je součástí Agrokromu. Do samotného aplikačního serveru bude proto potřeba doplnit funkce zajišťující komunikaci s P-Serverem, stažení vybraných částí systému na základě licencí, jejich korektní instalaci. Dále bude potřeba doplnit bezpečnostní prvky, týkající se aktualizací, korektního stažení částí systému a ověřování licencí. Především se bude jednat o funkce pro zabezpečení přenosu, zálohování uživatelských databází a částí systému a jejich obnovení do původního stavu v případě vzniku chyby.

Vlastní řešení funkcí doplňovaných do Agrokromu se musí držet zásadami, které byly dány při návrhu a implementaci aplikačního serveru. Proto bude nutné nastudovat dostupnou dokumentaci k aplikačnímu serveru a také se s touto aplikací seznámit jak na uživatelské, tak programátorské úrovni.

Projekt musí zahrnovat jak analýzu a návrh programového vybavení, tak uživatelskou a programátorskou dokumentaci.

2 Licenční a distribuční politiky software

V dnešní době, kdy s výpočetní technikou přichází do kontaktu většina lidí "vyspělé" společnosti, se velmi daří odvětví podnikání, zabývající se vývojem softwaru. Společnosti, jež se v tomto oboru pohybují, potřebují efektivní prostředky, kterými by mohli svoje díla propagovat, distribuovat, prodávat a především ochránit před zneužitím. Tato kapitola se bude zabývat především popisem distribuce a ochrany práv softwarového díla neboli programu.

Distribuční a licenční politiky softwaru jsou úzký pojem, který spolu silně souvisí. Na jejich základě dochází k rozdělení softwaru na několik skupin obr 2.1.1. Ty jsou často zaměňovány a spousta lidí neví, co jednotlivé pojmy znamenají.

2.1 Softwarová licence

Softwarová licence [1][3] je právní dokument (nástroj), která je připojena k počítačovému programu a ve které jsou uvedena práva a povinnosti smluvních stran (ve většině případů pouze práva a povinnosti nabyvatele licence). Softwarová licence je v ČR podmínkami licenční smlouvy. Nerozlišuje se smlouva a licence.

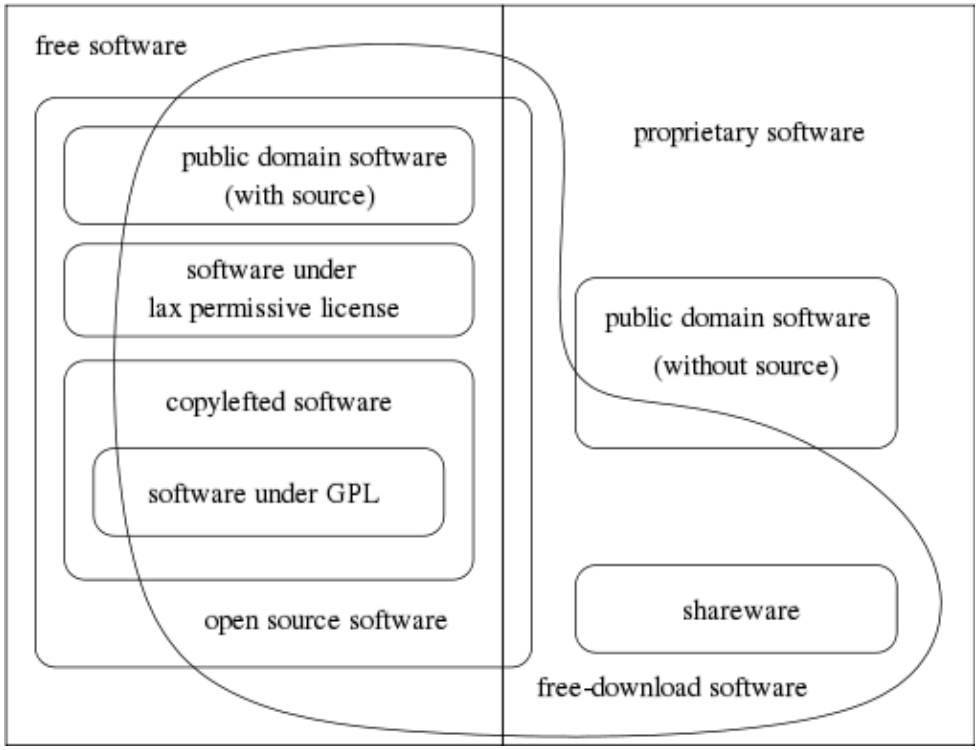
Mezi typické části, které bývají v licencích zmiňovány, jsou úpravy práv pro používání, kopírování, úpravy nebo distribuci software atd.

Aby měla licence nějaký smysl, musí se ustanovení v ní uvedené řídit platným zákonem. (V ČR to upravuje především Autorský zákon a Obchodní zákoník). Proto by ji měla vytvořit osoba s právním vzděláním s povědomím o problematice licencování a distribuce software. Často se totiž v licencích (především pro svobodný software ale i v jiných pro proprietární software) uvádějí body, jež jsou v rozporu se zákonem. Např.: Klauzule, ve které se autor zříká jakékoliv odpovědnosti za vzniklou škodu nebo se zříká samotného autorství, je dle [1] neplatná pro rozpor se zákonem. Lze to vidět i v licencích některých společností [5] jež výslovně uvádějí, že takovéto podmínky nejsou platné a jsou podle některých právních řádů neplatné.

2.1.1 Rozdělení software podle Free Software Foundation [2]

Free software - svobodný software - Svobodný software je software, který každému povoluje jej používat, kopírovat a rozšiřovat, buď přesně či se změnami, za poplatek nebo zdarma. Většina lidí si mylně spojuje význam svobodného software s tím, že je zdarma. Mezi hlavní body svobodného software patří:

1. Svoboda spouštět software za jakýmkoli účelem.
2. Svoboda studovat a měnit software ke svým potřebám.
3. Svoboda redistribuovat software.
4. Svoboda vylepšovat software a zveřejňovat tyto vylepšení ku prospěchu komunity.



Obr.: 2.1.1. : Dělení software podle Free Software Foundation zdroj GNU [2]

Open source software (OSS) – Software s otevřeným zdrojovým kódem – de facto je to totéž jako free software. Názory na to se sice různí ([1] – doporučuje používat OSS naproti tomu GNU doporučuje používat termín svobodný software) ale cíle obou hnutí jsou totožné. Rozdíl je především ve výkladu významu obou slovních spojení. Free software evokuje v uživateli, že software je zadarmo, což nemusí být a Open source, že musí mít otevřený zdrojový kód, což mít musí, ale nedává jasně najevo další podmínky svobodného software.

Public domain software[8] – Software bez vyhrazených práv – software bez nároku na nějakou právní ochranu. Autor dal dílo volně k dispozici a není nijak chráněno (v ČR to ovšem nelze, protože autorských práv se nelze vzdát) nebo jeho práva už vypršela (Typicky 70 let od smrti autora). Dílo, tedy v našem případě software, může používat každý jak chce ale podle autorského zákoníku si nesmí osobovat autorství ani užívat dílo tak, že by to snižovalo jeho hodnotu.

Copylefted software – Copyleft[4] je způsob ochrany, kdy se dílo odvozené od původního copyleftovaného díla musí řídit licencí původního díla. Copyleftové licence, jsou někdy nazývány jako virální licence – tzn., že pokud ve svém programu použijeme část, které je šířena pod copyleftovou licencí pak výsledné dílo musí být šířeno také pod copyleftovou licencí. Tímto je zaručeno, že budou dodrženy podmínky svobodného software. V případě, že by se střetly zájmy licencí dvou částí použitých v nějakém díle, pak musí být dílo šířeno pod copyleftovou licencí nebo vůbec. Mezi nejznámější Copyleftové licence patří licence GNU GPL od Richarda Stallmana.

Proprietary software – je nesvobodný software, jehož používání, kopírování, šíření a používání je zakázáno, pokud není autorem povoleno. Typicky se používá licence EULA.

Některé běžné licence:

GNU GPL [7] – GNU General Public Licence – je striktní copyleftová licence. Uvádí se, že až 65% svobodného software je šířeno pod touto licencí. V dnešní době se vyskytuje ve třech vydáních. Plně respektuje pravidla daná pro svobodný software.

GNU LGPL [9] – GNU Lesser General Public Licence – Je licence kompatibilní s GNU GLP. Oproti GPL se vyznačuje zeslabením copyleftového vlivu na dílo. To znamená, že dílo šířené pod touto licencí je možné za určitých podmínek šířit i pod jiným typem licence. Obvykle se používá na softwarové knihovny či jinné moduly.

GNU FDL [10] – GNU Free Documentation Licence – Je copyleftová licence používaná pro knihy, manuály a jiné dokumenty. Převážně se používá pro díla určená k výuce a poskytování informací.

EULA – End user license agreement – Licence pro koncového uživatele – která upřesňuje všechna práva nabyvatele a poskytovatele licence, kde si každý určí vlastní body licence. Nejčastější použití bývá u proprietárního software, ale lze pod ní šířit i jiné typy software. Například prohlížeč Firefox nebo Chrome (ikdyž jsou zdarma a jejich zdrojový kód je šířen pod open source licencí) jsou šířeny pod licencí EULA kvůli ochraně obchodní značky. Tedy každý si může stáhnout, upravit a šířit tyto aplikace dle libosti, ale nesmí takovéto upravené kopie šířit pod značkou Firefox nebo Chrome.

2.2 Distribuční politiky software

Klasický model fyzické realizace distribuce software, kdy se programy distribuovaly na pevných nosičích (disketách, CD, DVD) v tzv. krabicových vydáních se v dnešní době začíná opouštět. Může za to především fakt, že dostupnost a rozšířenost internetu rapidně stoupá, stejně jako stoupá kvalita připojení. Toto spolu s ušetřenými náklady na výrobu určuje, že distributoři přecházejí na model, kdy je software distribuován přes internet v elektronické formě. Dalším aspektem distribuce je i to, v jakém formátu se software k zákazníkovi dostane. (Demo, Shareware atd.) Pokud se k zákazníkovi dostane takovýto typ software, měla by (a většinou i bývá) k němu být připojena softwarová licence, která upravuje jeho další použití.

Shareware[2] – software, který je možné volně šířit, ale pro jeho trvalé využití je nutné se řídit jeho licenčními podmínkami a obvykle jej i zakoupit. Často bývá nějakým způsobem omezován – nejčastěji časově, pak hovoříme o trialware nebo funkčně crippleware. Shareware není svobodný software. Stává se, že aplikace po skončení období vyhrazeného na vyzkoušení je plně funkční (jediné omezení může být upozornění při instalaci, že program může být využíván po omezenou dobu) lidé přesto nebo právě proto porušují licenční podmínky. Takovou aplikací je například WinRar.

Demoware – neboli demoverze se označuje software, který je dostupný zdarma, ale je nějakým způsobem omezen. Nejčastějším způsobem omezení je omezená funkčnost oproti plné verzi programu. Význam je obdobný jako u shareware. Slouží k vyzkoušení programu.

Freeware[6] – opět se nejedná o svobodný software ale o proprietární software. Jde o software, který je možno používat zdarma. Často dochází k zaměňování pojmů freeware a free software (ve významu svobodný software) nebo open source software. Rozdíl je především v tom, že freeware je nutno šířit bezplatně oproti tomu svobodný software a open source software je možno šířit i za úplatu. Autor freeware si často ponechává některá práva. Typicky na modifikace (není zveřejněn zdrojový kód), případně šíření programu v komerční sféře za úplatu.

2.3 Licence Agrokromu

Cílem tohoto projektu není, navrhnout text licenční smlouvy. Jak již bylo v předchozích odstavcích řečeno, to by měla udělat osoba s právní vzděláním aby měla licence nějakou váhu. Částí projektu je sesbírat požadavky a zajistit technické a programové prostředky k podpoře nebo vynucení používání softwaru podle záměru majitele.

3 Agrokrom 6.0

Aplikace Agrokrom je systém pro poradce, agronomy a manažery v rostlinné výrobě. Protože předchozí verze už nevyhovovala z hlediska uživatelských požadavků a také použitých technologií, byl započat vývoj verze Agrokrom 6.0. Mezi požadavky na novou verzi byla i její škálovatelnost a tedy možnost užšího zaměření na konkrétní uživatele. Toho se dosáhlo zavedením systému modulů (addin, plugin). Moduly jsou samostatné programové balíčky, obsahující vždy sady funkcí z dané doménové oblasti. Zákazník si pak při koupi vybere k základní sadě pouze tu funkčnost, kterou ke svému podnikání potřebuje a sestaví si tak aplikaci takříkajíc „na míru“. Agrokrom je dále navržen jako klient/server aplikace. Tím bylo dosaženo toho, že více uživatelů v síti může pracovat nad jedněmi daty. Tomuto všemu se samozřejmě musí přizpůsobit systém licencování a distribuování aplikace.

3.1 Popis částí Agrokromu

Agrokrom se skládá ze tří samostatných programových částí, které se dají instalovat každá na samostatný počítač nebo v jednouživatelském režimu na jeden počítač.

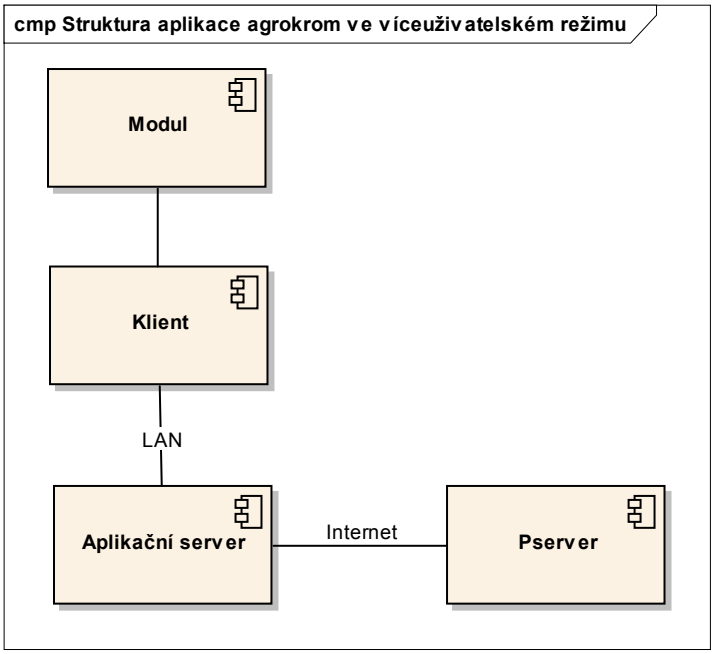
- **Aplikační server** – řídící aplikace Agrokromu. Zde by měl administrátor Agrokromu provádět hlavní nastavení a administraci aplikace a jejích klientů (pouze vybraná nastavení), která tak jsou odstíněna od běžných uživatelů. Aplikační server také bude monitorovat některé činnosti klientů a zajišťovat používání všech částí v rámci licenční politiky. Prostřednictvím aplikačního serveru bude mít možnost administrátor Agrokromu komunikovat s P-Serverem. Tím získá přístup k aktualizacím a některým funkcím pro rozšíření aplikace na základě objednávky.
- **Databázový server** – je aplikace třetí strany. Ze specifikace se bude jednat o databázový server Microsoft SQL Server 2005 Express. Na tuto část se samozřejmě vztahuje i příslušná licence od společnosti Microsoft.
- **Klient** – aplikace, která uživateli umožní vést běžnou evidenci.

Na obrázku 3.1.1 je znázorněna struktura aplikace Agrokrom 6.0 tak, jak ji je možno nainstalovat ve víceuživatelském režimu. Složená z Databázového serveru, Aplikačního serveru, 1-n klientů, kde každý z klientů obsahuje stejnou množinu modulů M_1-M_n.

Klient a moduly

Každý klient se skládá z modulů. Modul je část aplikace, která implementuje sadu funkcí zastřešující určitou doménovou oblast. (například modul pro vedení pozemkové evidence, modul ekonomických funkcí atd.). Klient bude v rámci základní licence obsahovat sadu minimálních nutných modulů, které budou potřebné k jeho chodu. Další moduly si bude moci zákazník do klienta dokoupit sám podle svých potřeb.

Každý modul se skládá z několika částí (především public a external). Systém byl navržen tak, aby konkrétní modul měl ve své kompetenci pouze určitou oblast databáze. Některé moduly ale využívají funkcí jiných modulů nebo potřebují data z té částí databáze, která nespadá do jejich kompetence. Proto se takovéto funkce, které využívají jiné části systému, umísťují do části modulu zvané external. V rámci licence ke konkrétnímu modulu zákazník získá i příslušnou část modulů, na kterých je závislý.



Obr. 3.1.1 – Struktura aplikace Agrokrom 6.0 ve víceuživatelském režimu

3.2 Licence Agrokromu 6.0

Jak už bylo v předchozí části dokumentu řečeno, zákazník si bude moci při objednání licence sestavit aplikaci takříkajíc na míru. Bude mu umožněno vybrat několik volitelných položek jako například doba platnosti licence, počet uživatelů, počet klientů, seznam modulů a podobně. Tato kapitola se zabývá některými požadavky kladenými na způsob licencování.

Platnost licence

Licence na Aplikaci Agrokrom 6.0 bude časová. Zákazník si bude moci sám vybrat, na jak dlouhou dobu si chce aplikaci objednat. Defaultní a minimální možná doba bude jeden rok. Licence začne platit ihned po jejím vystavení. K vystavení licence dojde až po jejím zaplacení. Poté si bude moci zákazník stáhnout aplikaci i svou licenci přes webové rozhraní a začít ihned používat. Další možností je, že mu bude aplikace zaslána poštou na CD.

Při offline objednání zde nastává problém s prodlevou na doručení aplikace a licence zákazníkovi. V takovýchto případech, se licence uměle prodlouží o dobu potřebnou na doručení zákazníkovi. Doba bude pravděpodobně 1 pracovní týden.

Po vypršení doby platnosti licence se musí aplikace Agrokrom 6.0 přepnout do tzv. režimu read, kdy uživatel aplikace bude moci pouze číst svá data (případně generovat tiskové sestavy). V režimu read budou muset být k dispozici také funkce pro objednání/prodloužení platnosti licence. Uživatel, během doby kdy nebude mít platnou licenci, ztratí možnost aktualizace aplikace. Aplikace by měla upozornit uživatele na možnost skončení platnosti aplikace v předstihu.

Pozn.: Aplikaci nelze úplně ukončit platnost, protože uživatelé mají povinnost evidovat svá data několik let a měli by k nim tedy mít přístup i po skončení platnosti licence.

Prodloužení platnosti licence

Prodloužením licence aplikace se myslí proces, kdy uživatel nebude v licenci nic měnit a pouze prodlouží platnost. V takovém případě bude moci licenci prodloužit a to před i po skončení platnosti původní licence a tím získat přístup k aktualizacím a zbylým funkcím. Prodloužení licence bude probíhat na základě objednávky prostřednictvím aplikačního serveru nebo webového rozhraní. (Případně offline objednání kdy na účet zákazníka vystaví objednávku pracovník P-Serveru.) Objednávka bude vystavena se stejnými náležitostmi jako při předchozím objednání. V případě jakýchkoliv změn údajů zahrnovaných do licence bude muset zákazník objednat novou licenci.

Stav jednotlivých částí systému se bude v čase měnit, bude se jednat hlavně o ceny jednotlivých modulů nebo případné slevy a omezení. Při prodlužování existujících licencí musí být celková cena vypočítána z aktuálně platných údajů. Odtud plyne i požadavek na vedení historie ceny a omezení jednotlivých částí licence tak aby bylo možno zpětně určit pro tyto údaje korektní hodnoty.

Seznam modulů zahrnutých v aplikaci Agrokrom

Při koupi aplikace si bude moci k základní sadě modulů vybrat z některých navíc zpoplatněných modulů a ty se mu ukotví v licenci. Zákazník si bude moci přidat do již existující licence nový modul, který bude moci od jeho zaplacení používat. Licence na modul bude časová a bude končit s platností aplikace Agrokrom. (Cena se vypočítá podle zbývající doby platnosti aplikace Agrokrom).

Vázaní aplikace na HW vybavení

Aplikace bude vázána na HW vybavení. Každý klient bude vázán na počítač, na kterém byl nainstalován. Toto musí probíhat plně automaticky při instalaci nebo při prvním spuštění. Administrátor Agrokromu pak bude muset mít k dispozici sadu funkcí, které mu umožní odregistrovat klienta, aby se nemusel obracet na poskytovatele licence v případě reinstalace

nebo poškození HW. Aplikační server pak bude kontrolovat počty klientů a uživatelů pracujících se systémem Agrokrom 6.0 na základě informací uvedených v licenčním souboru.

Jednotlivé položky licence

1. Licence bude vázána na firmu – ta bude vlastníkem licence.
2. V licenci bude uvedena doba platnosti a začátek platnosti licence. (Vybraná uživatelem)
3. Licence bude obsahovat počet klientů, které bude možno v rámci licence instalovat a používat. Defaultně jeden.
4. Licence bude také obsahovat počet uživatelů, kteří budou aplikaci používat. Do uživatelů se samozřejmě nebudou počítat speciální uživatelé jako správci, administrátoři jedná se pouze o uživatele klientských aplikací.
5. Licence bude obsahovat seznam modulů, které bude možno v rámci licence používat.
6. V licenci také musí být zahrnuta položka, která bude udávat typ licence vzhledem k velikosti firmy. Od toho se budou odvíjet omezení, kladené na aplikaci. Například uživatel bude moct založit pouze konkrétní počet pozemků nebo konkrétní celkovou výměru atd.

Zákazník v rámci licence získá vždy databázový server, aplikační server a počet klientů, který si zvolí při objednání aplikace. Taktéž získá jen ty moduly, které vybere v objednávce. Každý z dodaných klientů bude plně funkční a bude mít k dispozici všechny moduly, které zákazník objednal. Jednotlivé (různé) konfigurace modulů nebudou podporovány. Na Administrátorovi Agrokromu pak bude, aby přidělil práva k jednotlivým modulům svým uživatelům. Pokud dojde k doobjednání některého z modulů později, tento se nainstaluje do všech klientů a opět bude na administrátorovi, komu práva přidělí.

Licence aplikací třetích stran

Pokud Agrokrom bude využívat nějaké aplikace třetích stran, (které budou dodávané samostatně – např. mapové podklady) není možné nějak automaticky hlídat platnosti těchto licencí. Toto si bude muset pohlídat správce Agrokromu.

Licenční soubor

Licenční soubor, bude soubor s informacemi o licenci uživatele k produktu Agrokrom. Bude využíván k identifikaci, při komunikaci Agrokromu s P-Serverem a aktivaci aplikace Agrokrom. Taktéž bude využíván při aplikaci omezení na software, které budou z licence přímo vyplývat.

3.3 Distribuce Agrokromu 6.0

V rámci vyzkoušení programu by měla být přístupná na webových stránkách taktéž 30 denní demoverze Agrokromu distribuovaná jako shareware. Po uplynutí doby stanovené v licenci se bude muset uživatel řídit licenčními podmínkami a pro další použití software zakoupit.

Distribuce Agrokromu bude primárně probíhat prostřednictvím internetu on-line. V tomto případě se uživatel přihlásí prostřednictvím webového prohlížeče ke svému účtu na P-Serveru a tím získá přístup k možnosti objednání aplikace, nebo jejich částí. Dalším způsobem bude zpřístupnění některých funkcí pro objednání přímo v aplikačním serveru. Po zaplacení objednávky se automaticky k příslušnému uživatelskému účtu zpřístupní jednotlivé programové části aplikace. Poslední způsob, bude offline objednání prostřednictvím pracovníka podpory (telefonicky nebo jinou komunikací). V takovém případě správce P-Serveru založí na jméno zákazníka objednávku do P-Serveru a vygeneruje fakturu, po jejímž zaplacení P-Server automaticky vygeneruje příslušnou licenci. Licence i aplikace pak bude zaslána přímo jejímu novému majiteli poštou (nebo opět jinou cestou, která bude záležet na dohodě mezi zákazníkem a prodejcem).

3.4 Aktualizace

Aplikaci Agrokrom bude potřeba efektivně aktualizovat. K tomu musí být vytvořeny funkce, které tuto potřebu zajistí. Aktualizace by měly probíhat poloautomaticky nebo automaticky. Aplikační server by měl kontrolovat dostupnost aktualizací automaticky, ale tuto funkci by měl mít rovněž administrátor Agrokromu.

Systém aktualizací by měl mít obdobnou funkci jako například systém aktualizací ve Windows. Celý proces bude mít tři na sobě nezávislé fáze. Nejprve se ověří dostupnost nových aktualizací, poté dojde k jejich stažení a následně instalaci. Celý tento proces buď proběhne automaticky nebo poloautomaticky s tím že Administrátor Agrokromu bude muset potvrdit jednotlivé operace.

Před samotnou instalací aktualizací je nutno odpojit všechny připojené klienty. Každý klient při svém startu musí zkontrolovat, jestli se na Aplikačním serveru nenachází jeho nová verze. Uživatel klienta taktéž nesmí mít možnost pracovat na neaktualizovaném klientu, pokud na Aplikačním serveru budou nové aktualizace.

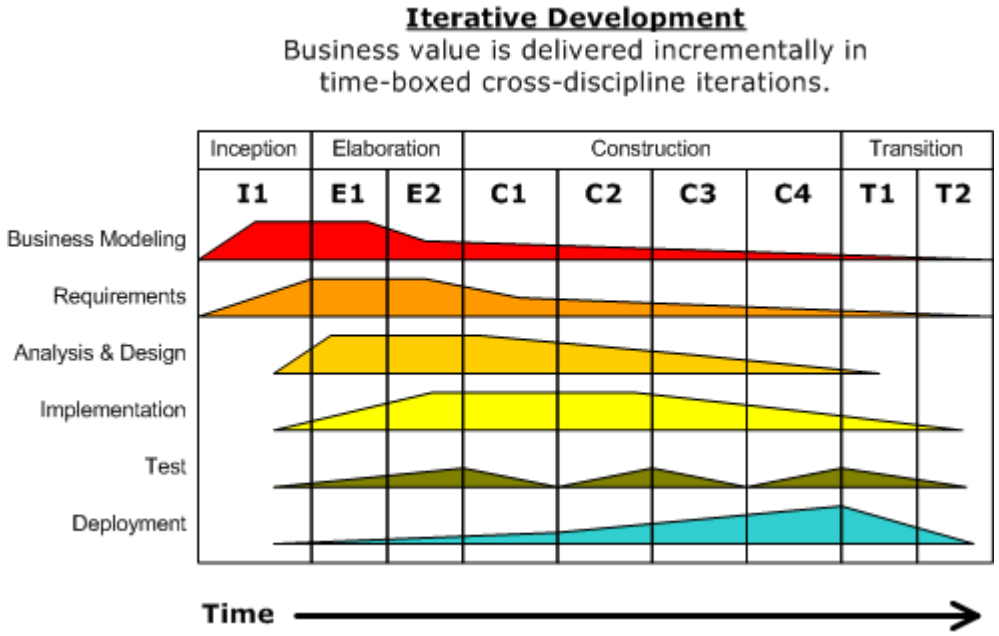
Konzistence databáze a aplikace

V systému budou existovat dva druhy aktualizací. Budou to aktualizace softwaru typicky knihovny, konfigurační soubory atd. a aktualizace databáze. Aktualizace softwaru nebudou inkrementální. Vždy se stáhne a aplikuje celý soubor. Aktualizace databáze ale budou muset být inkrementální kvůli datům v databázích uživatelů. Aktualizace databáze budou probíhat prostřednictvím skriptů, které připraví příslušný autor aktualizace. Ty budou následně spouštěny u klientů nad databází.

Pro případ neúspěchu při instalaci aktualizace musí před samotnou instalací dojít k záloze aplikace a databáze tak, aby bylo možné ji v případě chyby vrátit do původního stavu.Mezi verzemi aplikace a databáze bude existovat jasná vazba, tak aby bylo zřejmé, s jakou verzí aplikace je možné pracovat s danou verzí databáze. Verze aplikace bude jednoznačně určena verzemi všech svých částí. V systému se pak nesmí stát, aby uživatel používal verzi databáze, která nemá vazbu na verzi aplikace.

4 PServer

Tato kapitola se zabývá procesem vývoje PServeru. V jednotlivých podkapitolách popisuje specifikaci požadavků, analýzu, návrh, implementaci a nasazení PServeru. Jako metodika vývoje byly zvoleny procesy UP (Unified process) [12] a RUP (Rational Unified Proces). Z procesu RUP jsem použil některé dokumenty, které vhodně popisují problém řešení a zároveň budou sloužit jako projektová dokumentace. Jde o šablony dokument vize a software architecture dokument (SAD). Nezabýval jsem se všemi artefakty vývoje, ale pouze vybral ty, jež považuji za důležité nebo ty, jež přinášejí do projektu nějakou přidanou hodnotu.



Obr.: 4.1.1. – Fáze vývoje podle UP [15]

Vývoj je dle UP iterační, inkrementální a bude se skládat ze 4 hlavních iterací, kde v rámci každé fáze vývoje (inception, elaboration, construction, transition) proběhne 1 iterace. Obr 4.1 Celkový počet iterací je samozřejmě větší. Hlavně při specifikaci požadavků (fáze inception) probíhalo několik jednání, kde se požadavky neustále měnily, upravovaly a bylo zde několik menších iterací. Nemá ovšem smysl je nějak detailně popisovat, jelikož by pouze vznikaly mrtvé dokumenty.

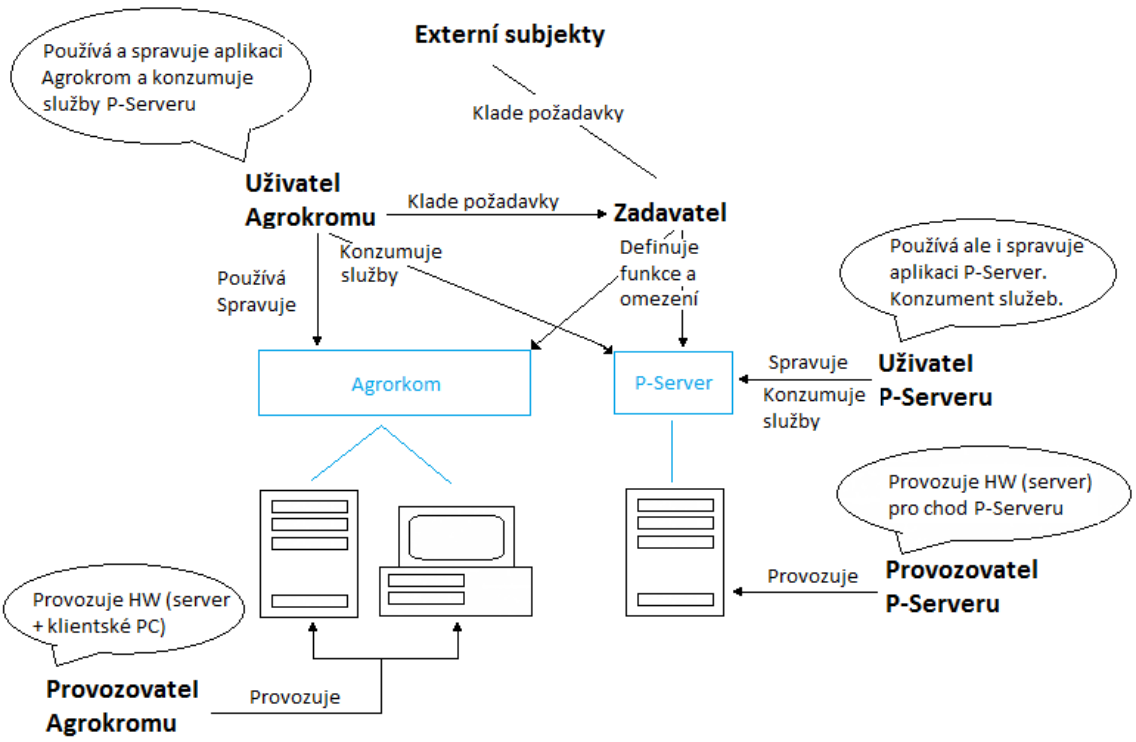
Aby byla dodržena souvislost následujícího textu, nebudu zde v rámci každé iterace rozebírat a popisovat jednotlivé aktivity vývoje, ty jsou detailněji rozepsány v dokumentaci. Popíšu zde aktivity a modely komplexně v rámci celého projektu (respektive na vybraných ukázkách) a na závěr uvedu seznam iterací, jejich cíle a zhodnocení dosažených výsledků.

4.1 Vize projektu

Účelem vize projektu a je sesbírat a identifikovat hlavní požadavky na P-Server od zadavatelů a uživatelů, určení účelu a zaměření systému, vymezení hranic a priorit navrhovanému řešení. Podkapitola poskytuje nadhled vyšší úrovně pro pochopení řešeného problému. Kompletní vizi projektu lze nalézt v projektové dokumentaci v souboru vize projektu[18].

4.1.1 Popis stakeholderů a uživatelů

Aplikace Agrokrom je systém pro poradce, agronomy a manažery v rostlinné výrobě. Jejich činnosti vysokou měrou ovlivňují orgány státní správy, které vydávají nařízení a vyhlášky, kterými se musí řídit. Majitel systému proto potřebuje efektivní způsob, jak reflektovat na požadavky úřadů (ale i uživatelů). Tato podkapitola popisuje okolí, uživatele a osoby, které nějakým způsobem ovlivňují systémy Agrokrom a P-Server.



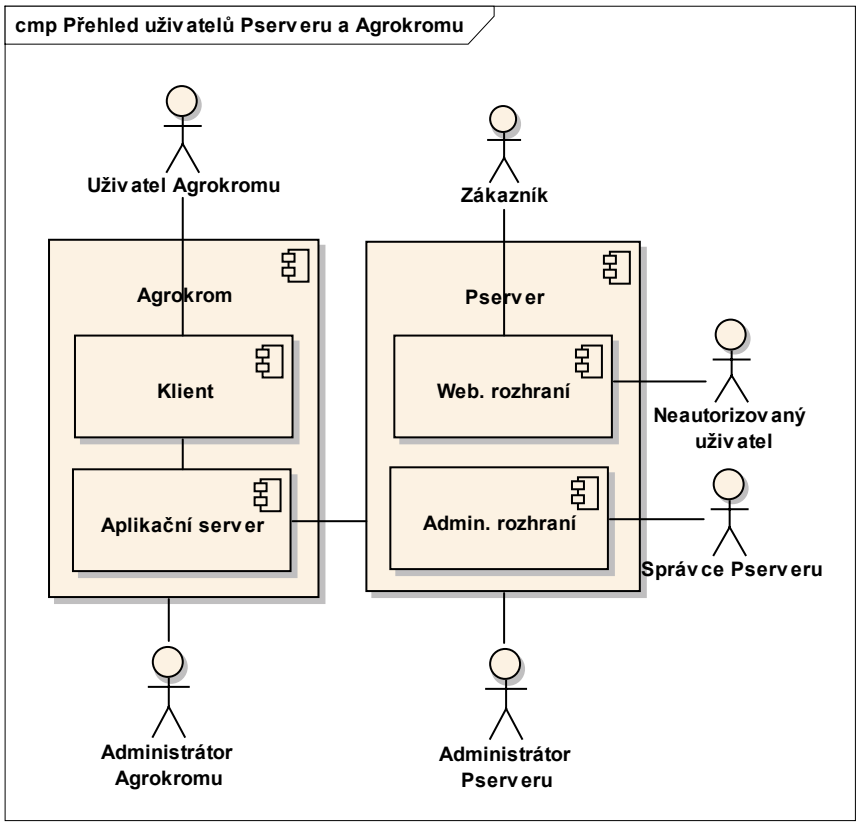
Obr: 4.1.1. – Popis stakeholderů a uživatelů

Přehled stakeholderů

Název	Reprezentuje	Role
Externí subjekty	Subjekty státní správy (různá ministerstva, kontrolní úřady atd., které mají vliv na podnikání firmy)	Ovlivňují systém vydáváním nových vyhlášek a nařízení, které musí uživatelé Agrokromu dodržovat a plnit. Tím nepřímo definuje specifikace na různé části systému a zadavatel systému na ně musí reflektovat.
Zadavatel	Zadavatele projektu (majitel P-Serveru a Agrokromu)	Definuje specifikace a omezení systému. Poskytuje prostředky na vývoj systému.
Provozovatel P-Serveru	Zemědělský výzkumný ústav Kroměříž, s.r.o.	Vlastní a provozuje HW vybavení, které bude potřeba k chodu P-Serveru.
Uživatel P-Serveru	Skupina uživatelů, konzumující služby P-Serveru ale i uživatelé starající se o chod aplikace.	- Administrátor P-Serveru, stará se o chod a zabezpečení aplikace P-Server - Správce P-Serveru bude osoba zodpovědná za práci s P-Serverem jako například vkládání dat. - Zákazník, užívá omezeného množství služeb P-Serveru prostřednictvím webové aplikace. - Neautorizovaný uživatel – náhodný přístup z webu.
Provozovatel Agrokromu	Provozovatel HW vybavení, na kterém bude instalována aplikace Agrokrom	Vlastní a provozuje vybavení, na kterém bude instalována aplikace Agrokrom.
Uživatel Agrokromu	Uživatelé aplikace Agrokrom a konzument služeb P-Serveru.	- Uživatel, který vede běžnou evidenci - Administrátor aplikace Agrokrom spravuje aplikační server a jeho prostřednictvím využívá služeb P-Serveru.

Přehled uživatelů P-Serveru

Tato sekce popisuje uživatele systému P-Server tak, jak vystupují ve svých rolích. Uživatelé (stakeholdeři také) byli z části převzati z dokumentace k Agrokromu [16], ale bylo nutné provést menší úpravy a rozšíření. Následující tabulka popisuje aktéry, kteří mohou se systémem P-Server nějakým způsobem pracovat. Uživatel Agrokromu jako takový nebude moci užívat služeb P-Serveru, protože z návrhu Agrokromu by neměl mít přístup k Aplikačnímu serveru. (Jedná se o více uživatelskou verzi. V jednouživatelské verzi splývá funkce Uživatele Agrokromu a Administrátora Agrokromu do jedné role.) Obr. 4.1.2 ukazuje uživatele obou systémů.



Obr: 4.1.2. – Přehled uživatelů P-Serveru a Agrokromu

Název	**Popis**	**Stakeholder**
Neautorizovaný uživatel	Speciální aktér, reprezentující náhodného uživatele přistupujícího k P-Serveru přes webovou aplikaci, bude moci pouze prohlížet ty části webové aplikace, které nebudou chráněny autorizací. Bude se jednat především o seznam a popis nabízených produktů.	Uživatel P-Serveru

Administrátor P-Serveru (PAdmin)	Instaluje a provozuje P-Server. Stará se o zajištění chodu aplikace a konfiguraci služeb P-Serveru.	Uživatel P-Serveru
Správce P-Serveru	Zajišťuje práci s P-Serverem, stará se o vkládání dat a úpravy informací k produktům. Komunikuje také se zákazníky a uživateli Agrokromu a zajišťuje podporu.	Uživatel P-Serveru
Zákazník	Jedná se o potencionálního zákazníka, jehož hlavním cílem je objednat aplikaci. Přistupuje ke službám P-Serveru prostřednictvím webového rozhraní. Může si objednat licenci Agrokromu nebo jeho částí.	Uživatel P-Serveru
Administrátor Agrokromu (AAdmin)	Instalace aplikace Agrokrom. Osoba zodpovědná za chod, nastavení a správu aplikací Agrokrom. Definuje práva k systému Agrokrom lidem ve firmě. Osoba s přístupem k Aplikačnímu serveru Agrokromu. Prostřednictvím aplikačního serveru může využít služeb P-Serveru.	Uživatel Agrokromu
Aplikační Server	Jednotlivé systémy aplikačních serverů, které budou spouštěny u zákazníka. Využívá některých služeb P-Serveru automaticky.	Aplikační Server

4.1.2 Uživatelské prostředí

Uživatelské prostředí se liší podle typu uživatele. Administrátor P-Serveru se podílí na chodu a nastavení aplikace P-Server. Ta bude instalována na serveru v prostředí výzkumného ústavu v Kroměříži. Systém bude provozován na operačním systému Windows Server 2003 a platformě .NET 3.5. Veškerá nastavení bude administrátor P-Serveru provádět přímo na serveru nebo prostřednictvím vzdálené plochy a lokální sítě výzkumného ústavu v Kroměříži. Server je umístěn v serverovně firmy, kam mají přístup pouze povolané osoby. Webové rozhraní P-Serveru se bude dát nainstalovat na stejný stroj jako samotný P-Server, nebo na jiný server. Webové rozhraní ke svému chodu bude potřebovat Internetovou informační službu (min. ve verzi 6.0) a ASP.NET. Správce P-Serveru bude osoba zodpovědná za práci s P-Serverem. Bude se starat o publikaci a úpravy dat (produktů) na P-Serveru a bude zajišťovat podporu zákazníkům. Správce P-Serveru bude přistupovat k P-Serveru z lokální sítě ve výzkumném ústavu v Kroměříži.

Potencionální zákazníci a neautorizovaní uživatelé, budou s aplikací komunikovat prostřednictvím webového rozhraní. Ke komunikaci s tímto rozhraním jim postačí běžný moderní webový prohlížeč (IE, Mozilla atd.). Pracovat se systémem P-Server budou moci odkudkoliv, kde získají přístup do sítě internet.

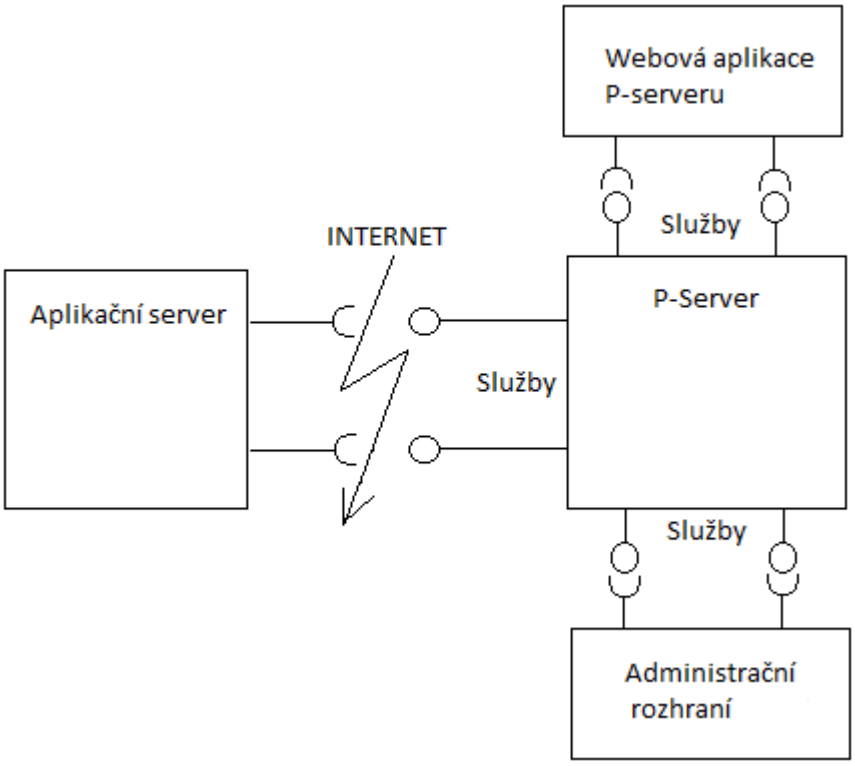
4.1.3 Klíčové cíle (potřeby) zadavatelů/uživatelů

Požadavek	Současné řešení	Navrhované řešení
Evidence uživatelských účtů	Není	Prostřednictvím webové aplikace nebo administračního rozhraní P-Serveru bude možno založit a spravovat uživatelský účet, který bude potřebný pro autorizovanou práci v systému.
Evidence firem	Není	Potřeba evidovat a přiřazovat firmy jednotlivým uživatelům. K firmám se pak budou vázat jednotlivé licence zakoupených produktů. Evidenci firem bude možno vést skrze webové rozhraní ale i přes administrační rozhraní P-Serveru.
Evidence licencí	Není	Potřeba vést evidenci licencí konkrétních firem, na základě které se bude poskytovat podpora majitelům těchto licencí. Prostřednictvím P-Serveru spravovat jednotlivé licence.
Objednání produktů	Není. V současnosti není pro verzi 6.0 řešení. V předchozích verzích probíhalo objednávání aplikace a jejích částí offline. (telefon)	Objednávat bude možno skrze webové rozhraní. Do aplikačního serveru budou doplněny některé funkce pro práci s licencí. P-Server bude mít mechanismy pro vystavování faktur a ověřování plateb a informování klientů.
Distribuce produktů	Není. Existuje pouze informační webová stránka a poté si uživatel musí objednat telefonicky. Aplikace je pak doručena poštou.	Administrátor P-Serveru musí mít k dispozici nástroje pro vystavení produktů, jejich popisu a souvisejících licenčních politik, které bude moci zákazník získat přes internet.
Distribuce aktualizací	Není. Momentálně je možno pouze rozeslat na vyžádání poštou nebo elektronickou poštou. Což musí provést zaměstnanec Vukromu	Administrátor P-Serveru bude vystavovat průběžné aktualizace (patche) pro jednotlivé produkty. Uživatelé poté budou moci využít služeb P-Serveru a instalovat tyto aktualizace do Agrokromu.
Ověřování licencí	Není. V současnou existuje mechanismus, kdy uživatel vygeneruje ze svého sw kód, který je poté zaslán pracovníkovi vukromu, který jej ověří.	Musí se navrhnout a do Agrokromu implementovat mechanismy, které uživateli dovolí používat aplikaci pouze s licenčním ujednáním.

Zálohování	Není	Jak na P-Serveru tak na App serveru bude potřeba dodělat jednoduchého správce, který bude umět zálohovat jednotlivé části systému a poté je uvést do původního stavu. Správce musí být provázán na aktualizace tak, aby bylo možné provádět bezpečné aktualizování aplikace Agrokrom.

4.1.4 Náhled na systém P-Server

Jak je patrné z předchozího textu, P-Server bude serverová aplikace běžící jako služba. Bude disponovat administračním rozhraním pro správu aplikace. Svému okolí bude vystavovat do sítě a internetu služby, které budou plnit požadavky zadavatele. (správa zákaznických účtů, distribuce aktualizací, distribuce licencí, objednávání atd.). Ke konzumaci služeb P-Serveru bude potřeba využít „klienta“. V této roli bude vystupovat aplikační server nebo webový prohlížeč. To jaké funkce budou dostupné u konkrétních typů klientů, bude upřesněno později v analýze případu užití.



Obr: 4.1.3 – Náhled na systém PServer

Licencování a instalace PServeru

P-Server je určen Zemědělskému výzkumnému ústavu Kroměříž, s.r.o. a veškerá práva na něj budou majetkem této firmy. Produkt se bude skládat ze samotného P-Serveru, databázového serveru (licence třetí strany) a webové aplikace. Prodej tohoto systému se nepředpokládá.

Licence na programové vybavení doplňované do aplikačního serveru aplikace Agrokrom se budou odvíjet od licenční politiky, která se bude aplikovat na prodej systému Agrokrom jako takového, nebo na jeho části. Cílem projektu je navrhnou i tyto licenční politiky. Návrh konkrétního způsobu licenčních politik pro Agrokrom viz. Kapitola 3.2

Instalace bude probíhat z instalačních médií, samotný P-Server musí být možno nainstalovat na server jednoduchou instalační aplikací. Tato aplikace musí zahrnovat minimálně instalaci databáze a P-Serveru a jejich konfiguraci. Instalaci webového rozhraní bude pravděpodobně potřeba provést zvlášť. Pokud to bude možné, bylo by dobré ji taktéž zahrnout do instalační aplikace.

Části aplikace, které budou doplněny do aplikačního serveru, budou instalovány zároveň s Agrokromem. Instalace Agrokromu není předmětem tohoto projektu.

4.1.5 Funkční a nefunkční požadavky

Tato podkapitola by měla pro úplnost obsahovat stručný seznam funkcí a zodpovědností. Nicméně zde tento seznam uvádět nebudu. Pro vytvoření představy o tom jaké budou funkční požadavky, postačí podkapitola 4.1.5 Klíčové cíle (potřeby) zadavatelů/uživatelů. Zjednodušený seznam funkcí lze nalézt v [18] v dokumentu vize a kompletní seznamy funkcí i s popisem lze nalézt v dokumentu specifikace softwarových požadavků.

Nefunkční požadavky:

Funkcionalita

Systém podporuje všechny funkce vymezené v modelu případů užití.

Dostupnost

Protože systém P-Server bude sloužit jako podpora zákazníků, jeho služby musí být dostupné 24 hodin denně a 7 dní v týdnu. Dostupný musí být jak přes webové rozhraní, tak přes rozhraní Aplikačního serveru.

Použitelnost

Systém by měl být intuitivně použitelný.

Bezpečnost

V návrhu a implementaci systému bude potřeba zahrnout určité bezpečnostní prvky. Především půjde o zabezpečení přenosu komunikace a dat mezi P-Serverem a jeho klienty, aby nedošlo k odposlouchávání nebo zcizení těchto dat.

Dále musí být zajištěna ochrana uživatelských údajů na P-Serveru (jakási garance toho, že informace o uživatelích a firmách nebudou dány nikomu k dispozici a budou k nim mít přístup

pouze povolané osoby a to jen v rámci určitých práv). Toto bude potřeba vyřešit jak v rámci zabezpečení aplikace, implementací vhodných technologií, tak v rámci dohody mezi zákazníkem a poskytovatelem, která by měla být k dispozici při registraci do systému. (Podmínky užívání + ustanovení o ochraně osobních údajů).

Systémové požadavky

Systémové požadavky se budou odvíjet od typu použité aplikace a jejích součástí, především musí stanice splňovat požadavky na .NET framework[1] a databázový server SQL Server 2005[2]. P-Server pak ke své činnosti bude využívat i služeb webovského serveru.

SW Požadavky na aplikaci P-Server a její části**:**

- P-Server bude využívat databázi MS SQL Server 2005 Express Edition nebo vyšší.
- P-Server poběží nad frameworkem .NET 3.5 nebo vyšším.
- Webovský server – Internetová Informační Služba IIS 6.0
- Požadavky na webového klienta – jakýkoliv moderní webový prohlížeč (IE, Mozzila, Opera, …)

Minimální HW požadavky odvozené od požadavků pro .NET framework 3.5 a SQL Server 2005 Express:

- **RAM:** 192 MB (*Minimum*) / 512 MB (*Doporučeno*)
- **Procesor:** 600MHz (*Minimum*) / 1GHz (*Doporučeno*)
- **Disk:** 600MB pro Databázový server + 500MB pro .NET Framework
- **OS:** Windows 2003, Windows Server 2008, Windows Vista, Windows XP

Instalační a konfigurační příručka

Instalační a konfigurační příručka bude určena především administrátorům systému. Bude popisovat Instalaci jednotlivých komponent P-Serveru a jejich podrobné nastavení.

Součástí bude i uživatelská dokumentace, která bude obsahovat jednak uživatelský popis funkcí, které se budou doplňovat do aplikačního serveru a dále uživatelský manuál pro administrátora a správce P-Serveru.

Dokumentace projektu

Bude obsahovat celkovou dokumentaci návrhu a analýzy systému a dále popis jednotlivých funkcí programu.

---

[1] Požadavky .NET http://www.microsoft.com/downloads/details.aspx?FamilyID=AB99342F-5D1A-413D-8319-81DA479AB0D7&displaylang=en

[2] Požadavky SQL Server - http://www.microsoft.com/Sqlserver/2005/en/us/system-requirements.aspx

4.2 Specifikace požadavků

Základní specifikace požadavků a směr projektu již byly nastíněny v dokumentu vize. V této podkapitole se zabývám modelováním požadavků a případů užití a dále sledováním požadavků k případům užití. Příklady jsou demonstrovány na části modelu pro evidenci uživatelských účtů v systému PServer.

4.2.1 Model požadavků

Často bývají softwarové požadavky určeny pouze jako dokument obsahující text zevrubně popisující co by měl software dělat. Význam a hodnota takovýchto dokumentů je však přinejmenším sporná. Dá se na jejich základě udělat obrázek o tom co má software dělat ale pro další práci nejsou příliš vhodné. Tyto dokumenty jsou vlastně spíše vizemi.

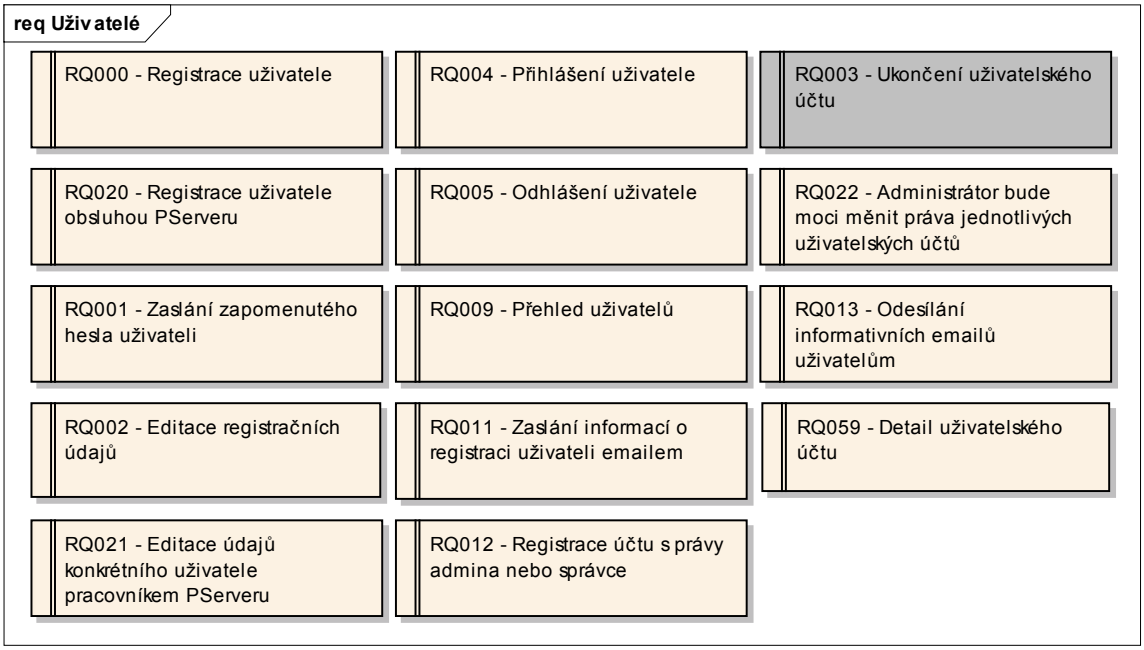
Obr.: 4.2.1 – Rozdělení požadavků PServeru do kategorií

Lepším přístupem je na základě takovéto vize sestavit model požadavků. Model požadavků není nic jiného než dokument obsahující číslovaný seznam požadavků i s jejich popisem v nějakém předem daném formátu. Modelovací nástroje přidávají i další parametry, jako stav požadavku (přijatý, zamítnutý) atd. a tvoří taxonomii požadavků dle potřebných kritérií. Požadavky jsem pak rozčlenil do skupin (funkční, nefunkční, a dále podle logicky souvisejících vlastností) viz

obrázek 4.2.1. Následující část obsahuje fragment modelu požadavků. Celý dokument je dostupný v [18] v souboru specifikace softwarových požadavků.

Model požadavků – požadavky na uživatelskou evidenci

V této podkapitole se nachází část modelu, popisující požadavky pro evidenci uživatelských účtů. Obrázek 4.2.2. ukazuje vzhled modelu požadavků v Case Nástroji Enterprise Architect. Je to pouze ukázka části modelu. Z obrázku je vidět i zamítnutý požadavek RQ003, který má přiřazen atribut Rejected.



Obr.: 4.2.2 – Model požadavků – požadavky na uživatelskou evidenci

RQ000 - Registrace uživatele	PServer bude umět provést registraci uživatele do systému se zákaznickými právy. Uživatelé se budou registrovat sami přes webové rozhraní.
RQ001 - Zaslání zapomenutého hesla uživateli	Každý uživatel si bude moci poslat na svůj registrační email nové přihlašovací údaje. Tedy staré přihlašovací jméno a systémem nově vygenerované heslo.
RQ002 - Editace registračních údajů	Každý uživatel přihlášený do systému přes webovou aplikaci nebo přes podnikové rozhraní bude moci editovat některé své registrační údaje.
RQ003 - Ukončení uživatelského účtu	Na žádost zákazníka bude administrátor provádět ukončení platnosti uživatelského účtu. Uživatel

	musí být o provedených akcích informován.
RQ004 - Přihlášení uživatele	Přihlášení uživatele do systému. Každý kdo bude nějakým způsobem se systémem PServer pracovat bude muset být přihlášen.
RQ005 - Odhlášení uživatele	Odhlášení ze systému PServer
RQ009 - Přehled uživatelů	Pracovník pracující s PServerem bude potřebovat funkci, která mu zobrazí přehled všech registrovaných uživatelů v systému a odtud bude provádět nad jednotlivými účty další operace.
RQ011 - Zaslání informací o registraci uživateli emailem	Systém po registraci automaticky pošle email s registračními údaji uživateli
RQ012 - Registrace účtu s právy administrátora nebo správce	Administrátor bude moci registrovat nový účet administrátora nebo správce, který bude pracovat jako obsluha (pracovník podpory) systému PServer.
RQ013 - Odesílání informativních emailů uživatelům	Pracovník PServeru bude potřebovat funkci, která bude informovat uživatele při některých změnách prováděných Správcem v Pserveru.
RQ020 - Registrace uživatele obsluhou PServeru	V případě, že uživatel nebude mít dostupné připojení k internetu, může kontaktovat pracovníka PServeru, který provede registraci účtu místo něj.
RQ021 - Editace údajů konkrétního uživatele pracovníkem PServeru	Pracovník PServeru bude potřebovat funkce pro editaci údajů vybraného uživatele.
RQ022 - Administrátor bude moci měnit práva jednotlivých uživatelských účtů	Administrátor vybere konkrétního uživatele a tomu změní práva do systému.
RQ059 - Detail uživatelského účtu	Správce i zákazník budou mít k dispozici funkci pro zobrazení detailu uživatelského účtu.

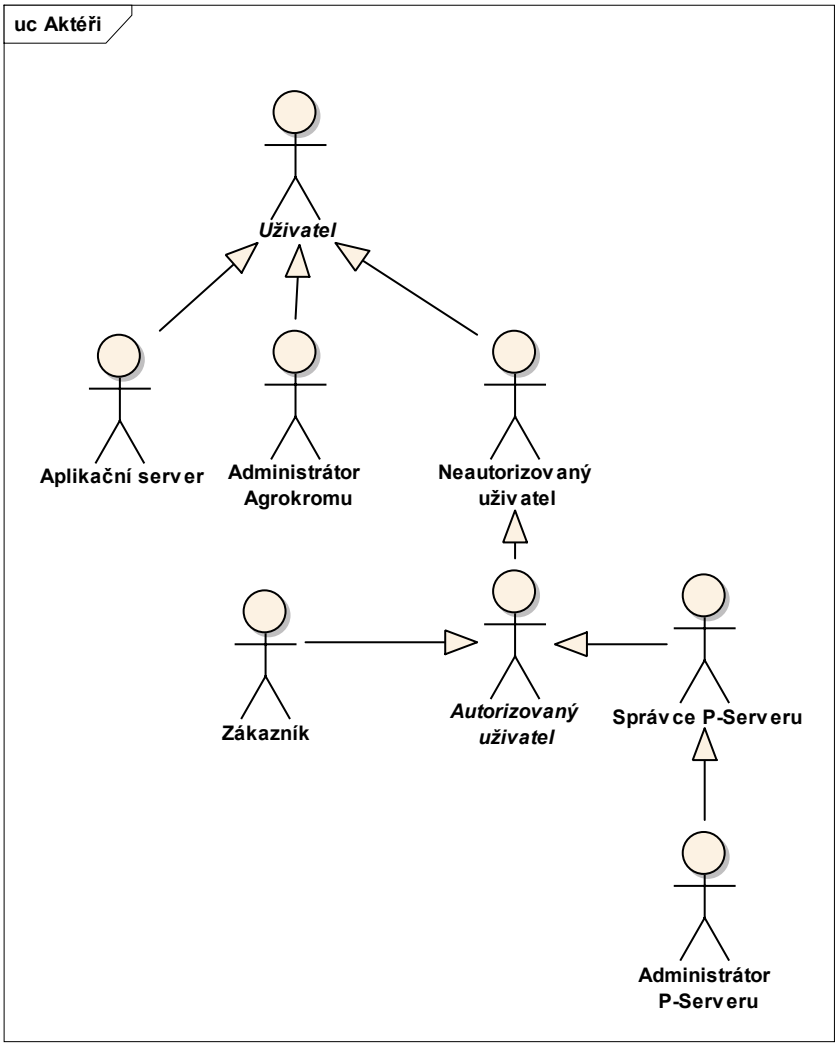
4.2.2 Model případů užití

Model případů užití navazuje na model požadavků a společně tvoří specifikaci softwarových požadavků. V UML neexistuje nástroj, kterým by bylo možno modelovat požadavky jinak než pomocí případů užití. Vstupními artefaky pro modelování případů užití je model požadavků, dokument vize a případně business model(obchodní model). Modelování případů užití má několik aktivit jimiž jsou [12]

- Vymezení hranic systému
- Nalezení aktérů
- Nalezení případů užití a scénářů případů užití
- Nalezení vazeb mezi aktéry a případy užití

Aktéři

Aktéry v tomto případě můžeme převzít z dokumnetu vize, kde již byli někteří hlavní aktéři nalezeni a popsáni viz kapitola 4.1.1. Jelikož potomek aktéra dědí všechny vazby na případy užití svého předka, byli do modelu přidáni další aktéři. Tito abstraktní aktéři mají za úkol zpřehlednit model případů užití a nejsou to reální aktéři, kteří budou se systémem pracovat. Na obrázku 4.2.3 je znázorněna hierarchii aktérů.



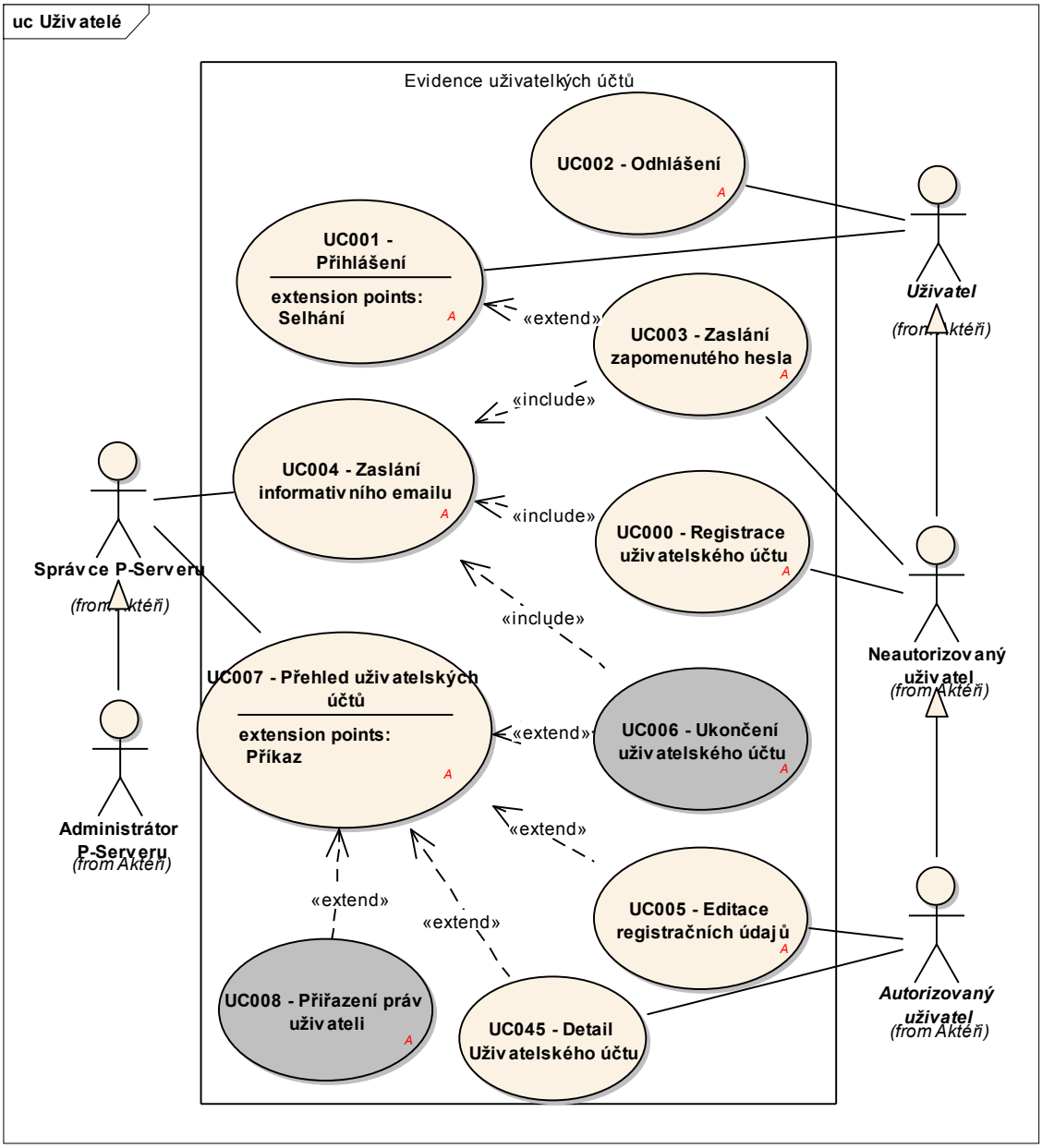
Obr.: 4.2.3 – Hierarchie aktérů

Aktéři, o které byla rozšířena vize projektu:

- Uživatel – Abstraktní uživatel reprezentující všechny aktéry (uživatele a systémy) v systému P-Server. Všichni aktéři se budou moci v systému nějakým způsobem autentizovat a autorizovat. Je to vlastně neautorizovaný uživatel. Nicméně kvůli tomu, že neautorizovaný uživatel přistupující přes některé z rozhraní PServeru bude mít navíc

některé funkce oproti aktérům přistupujícím ze strany rozhraní Agrokromu musel být zaveden tento aktér zvlášť.

- Autorizovaný uživatel – je abstraktní aktér zavedený pro přehlednost. Reprezentuje všechny autorizované aktéry (Zákazníka, Správce i Administrátora Agrokromu), přes webové rozhraní nebo administrační rozhraní PServeru.
- Aplikační server – se bude automaticky připojovat k PServeru pro kontrolu aktualizací. Proto bude také vystupovat jako aktér.



4.2.4. – Diagram případů užití pro evidenci uživatelských účtů

Nalezení případů užití a scénářů případů užití

Po identifikaci aktérů je potřeba nalézt případy užití. Případ užití je posloupnost akcí, které aktér v systému provádí v rámci dosažení nějakého cíle. Případ užití taktéž určuje, co aktér od systému očekává. Diagram je důležitý především pro komunikaci se zákazníky a scénář jednoznačně popisuje jednotlivé akce případu užití. Na obrázku 4.2.4 je zobrazen diagram případů užití, pro evidenci uživatelských účtů. Stejně jako v modelu požadavků jsou i zde šedě vybarveny případy užití s atributem rejected, tedy zamítnuté.

Ukázka scénáře případu užití

Následuje ukázka scénáře vybraného případu užití UC005 – Editace registračních údajů. Primárním úkolem scénáře je editace vlastních registračních údajů samotným uživatelem. Nicméně tento případ užití navíc realizuje ve speciálním režimu scénář, kdy administrátor PServeru při editaci změní práva uživatele. Hlavními aktéry jsou tedy všichni, kteří jsou přihlášeni (určeno abstraktním aktérem autorizovaný uživatel) a administrátor PServeru. Scénář obsahuje i možné alternativní scénáře, kdy například při změně registračních údajů uživatel nevhodně vyplní některá data a systém jej na to upozorní. Dalším alternativním scénářem je i to, když si uživatel během editace svoje konání rozmyslí a akci zruší.

Protože UML nijak neupravuje ani neobsahuje šablonu pro scénáře případů užití, použil sem pro popis scénářů případů užití v PServeru šablonu scénáře uvedenou v [12].

ID:	**UC005**
Název:	**Editace registračních údajů**
Popis:	Autorizovaný uživatel provádí editaci registračních údajů vlastního účtu
Primární aktéři:	Autorizovaný uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	Autorizovaný uživatel je v systému přihlášen
Hlavní scénář:	1. Scénář je spuštěn autorizovaným uživatelem, když vybere změnit registrační údaje 2. Autorizovaný uživatel změní hodnoty registračních údajů 3. Dokud jsou změněné registrační údaje neplatné 3.1. Systém žádá autorizovaného uživatele, aby zadal platné registrační údaje 3.2. Systém ověří editované registrační údaje 4. Systém upraví registrační údaje uživatele 5. Systém zobrazí autorizovanému uživateli zprávu o provedených změnách
Výstupní podmínky:	Registrační údaje uživatelského účtu byly upraveny
Alternativní scénáře:	Neplatné registrační údaje Storno Změna práv uživatele

ID:	**UC005.01**
Název:	**Editace registračních údajů: Neplatné registrační údaje**
Popis:	Systém informuje autorizovaného uživatele, že údaje nevyhovují podmínkám systému.
Primární aktéři:	Autorizovaný uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	Autorizovaný uživatel změnil nevhodně registrační údaje
Alternativní scénář:	1. Alternativní scénář začíná krokem 3.2 hlavního scénáře 2. Systém upozorní autorizovaného uživatele na špatně změněné registrační údaje
Výstupní podmínky:	Žádné

ID:	**UC005.02**
Název:	**Editace registračních údajů: Storno**
Popis:	Autorizovaný uživatel stornuje editaci registračních údajů uživatelského účtu
Primární aktéři:	Autorizovaný uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	Žádné
Alternativní scénář:	1. Alternativní scénář začíná kdykoli během hlavního scénáře 2. Autorizovaný uživatel stornuje editaci registračních údajů
Výstupní podmínky:	Registrační údaje uživatelského účtu zůstanou nezměněny v původním stavu

ID:	**UC005.03**
Název:	**Editace registračních údajů: Změna práv uživatele**
Popis:	Administrátor PServeru může provést editaci registračních údajů se změnou práv uživatele
Primární aktéři:	Administrátor PServeru
Vedlejší aktéři:	Žádní
Vstupní podmínky:	Administrátor PServeru je přihlášen
Alternativní scénář:	1. Alternativní scénář začína krokem 3 hlavního scénáře 2. Administrátor PServeru vybere typ role uživatele 3. Scénář pokračuje krokem 4, hlavního scénáře
Výstupní podmínky:	Registrační údaje uživatelského účtu byly upraveny Role uživatele byly upraveny.

4.2.3 Sledování požadavků k případům užití

Z obrázku 4.2.4 je patrné, že dva případy užití byly zamítnuty (jsou to ty vybarveny šedě). Je to proto, že případ užití UC006 – Ukončení uživatelského účtu byl zamítnut na základě zamítnutí požadavku RQ003 – Ukončení uživatelského účtu, který již nebyl potřeba. Protože případy užití realizují požadavky, pak tento již dále neměl smysl. Další případ užití UC008 - Přiřazení práv uživateli byl zamítnut na základě zjištění, že tento požadavek je již realizován jiným případem užití a to UC005 – Editace registračních údajů. Jejich společný požadavek RQ022 zůstal zachován. Pro sledování požadavků a případů užití jsem použil užitečný nástroj matice sledovatelnosti [12], díky které je možno propojit model požadavků a model případů užití tak, aby bylo jasné, které požadavky jsou nebo nejsou realizovány, či které případy užití realizují konkrétní požadavky. Díky této matici jsem mohl kontrolovat, jestli některé požadavky nepostrádají svou realizaci v případech užit a opačně. Obrázek 4.2.5 ukazuje propojení obou modelů. Je zde jasně vidět, že zde jsou vazby M:N. To znamená, že jeden požadavek může být realizován více případy užití a opačně.

	Uživatelé::UC000 - Registra	Uživatelé::UC001 - Přihláše	Uživatelé::UC002 - Odhláše	Uživatelé::UC003 - Zaslání z	Uživatelé::UC004 - Zaslání i	Uživatelé::UC005 - Editace r	Uživatelé::UC006 - Ukončer	Uživatelé::UC007 - Přehled	Uživatelé::UC008 - Přiřazen	Uživatelé::UC045 - Detail Už
Uživatelé::RQ000 - Registra...	←				←					
Uživatelé::RQ001 - Zaslání ...				←	←					
Uživatelé::RQ002 - Editace ...						←				
Uživatelé::RQ003 - Ukonče...					←		←	←		
Uživatelé::RQ004 - Přihláše...		←								
Uživatelé::RQ005 - Odhláše...			←							
Uživatelé::RQ009 - Přehled ...								←		
Uživatelé::RQ011 - Zaslání i...	←				←					
Uživatelé::RQ012 - Registra...	←							←		
Uživatelé::RQ013 - Odesílá...					←					
Uživatelé::RQ020 - Registra...	←				←					
Uživatelé::RQ021 - Editace ...						←		←		
Uživatelé::RQ022 - Adminis...						←		←		
Uživatelé::RQ059 - Detail u...										←

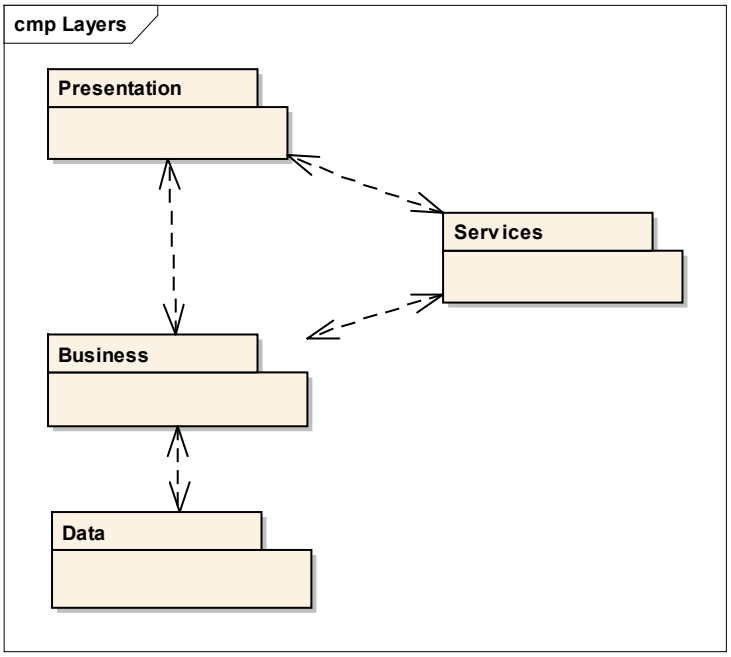
Obr.: 4.2.5. – Matice sledovatelnosti

4.3 Analýza a návrh

Tato kapitola prezentuje architekturu systému, jako soubor pohledů podle SAD viz [18] – Software architecture document. Všechny pohledy jsou modelovány pomocí jazyka UML. Základní architektura musí umožňovat splnit všechny hlavní požadavky uživatelů a mít schopnost oddělit řídící aplikaci od databázových serverů a od klientů. Také musí být schopna napojení několika druhů rozhraní především, klasického webové a desktop rozhraní a umožnit přístup aplikačním serverům prostřednictvím služeb.

4.3.1 Architektura systému

Tato část poskytuje bližší pohled na celkovou architekturu. Byla zvolena 4-vrstvá architektura viz. obr. 4.3.1, která umožní oddělit prezentační vrstvu od aplikační logiky a komponent pro přístup k datům. Každá vrstva bude moci komunikovat bezprostředně pouze s nižší vrstvou. Na obr 4.3.2 je detailně zobrazen přehled vrstev a jednotlivých komponent v nich zastoupených. Speciální vrstvu pak tvoří vrstva služeb, ta bude poskytovat funkcionalitu daným uživatelským rozhraním. Celý systém je zastřešen .NET frameworkem 3.5 a jeho komponentami jak ukazuje obr. 4.3.3.



Obr.:4.3.1 – Přehled vrstev systému PServer

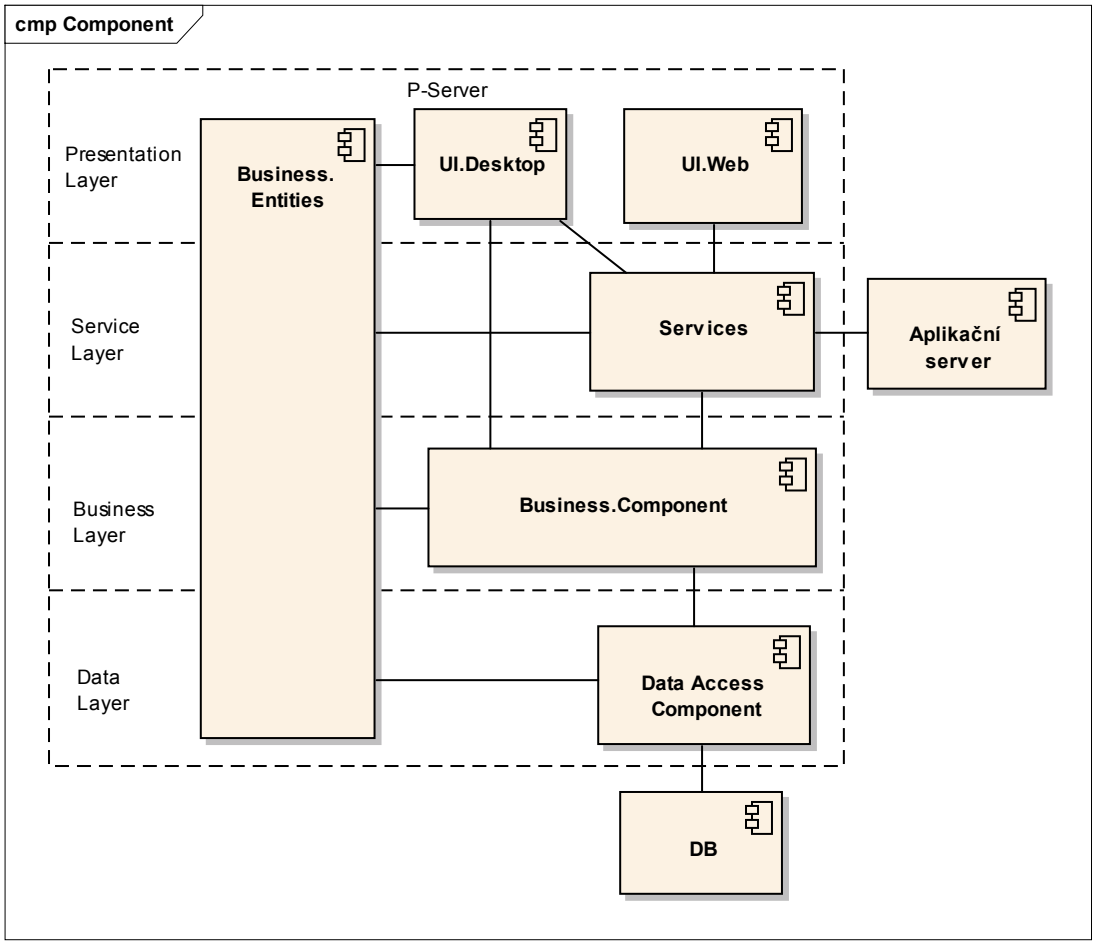
Prezentační vrstva

Poskytuje ovládací prvky systému. Musí obsahovat komponenty pro vytvoření webového rozhraní a také, desktop klienta určeného pro běh v místní síti. S výhodou lze využít možnosti ASP.NET pro webové rozhraní a Windows forms pro desktopové rozhraní.

- *UI.Desktop* – komponenta reprezentující desktopové administrační rozhraní PServeru. Zde budou veškeré ovládací prvky určené administrátorům.
- *UI.Web* – je webové rozhraní obchodní stránky. Webové rozhraní bude se zbytkem aplikace komunikovat pouze skrze servisní vrstvu. Výjimkou budou některé komponenty business vrstvy.

Service Layer

Protože celý systém bude využíván různými druhy klientů, bylo vhodné zařadit mezi prezentační a aplikační vrstvu ještě vrstvu servisní. Ta umožní dokonalé oddělení aplikační a prezentační logiky a navíc nechá otevřené možnosti například pro další rozšíření nebo připojení dalších systémů. Pro vytvoření servisní vrstvy lze využít komponent z Windows Communication Foundationn (WCF). K servisní vrstvě budou také připojovány jednotlivé aplikační servery zákazníků.



Obr.: 4.3.2 – Přehled komponent v jednotlivých vrstvách

Business Layer

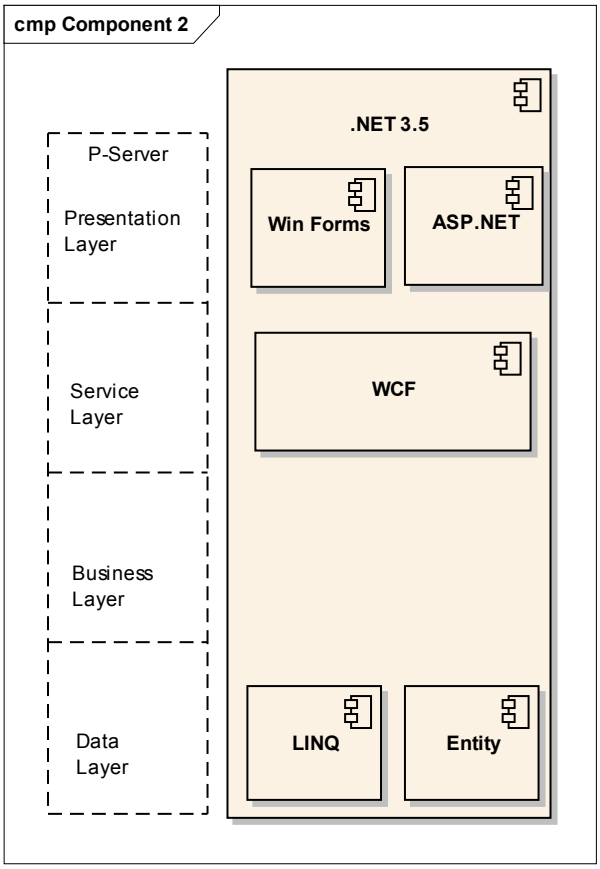
Vrstva obsahující aplikační logiku systému v komponentě *business componetnt*. Vrstva dále bude obsahovat speciální komponentu *business entities*, která nepatří přímo do business vrstvy ale prolíná všemi vrstvami. *Business entities* bude obsahovat jednotlivé business entity (například uživatele atd.) které si budou ostatní vrstvy mezi sebou předávat.

Datová vrstva

Bude obsahovat pouze jednu komponentu *data access component*. Ta bude zapouzdřovat pouze funkčnost sloužící k přístupu a práci s databází. Umožní komunikaci s komponentou DB, která reprezentuje na obrázku databázový server SQL. Funkčnost bude zprostředkována pomocí frameworků LINQ a Entity z .NET 3.5.

Databáze – DB

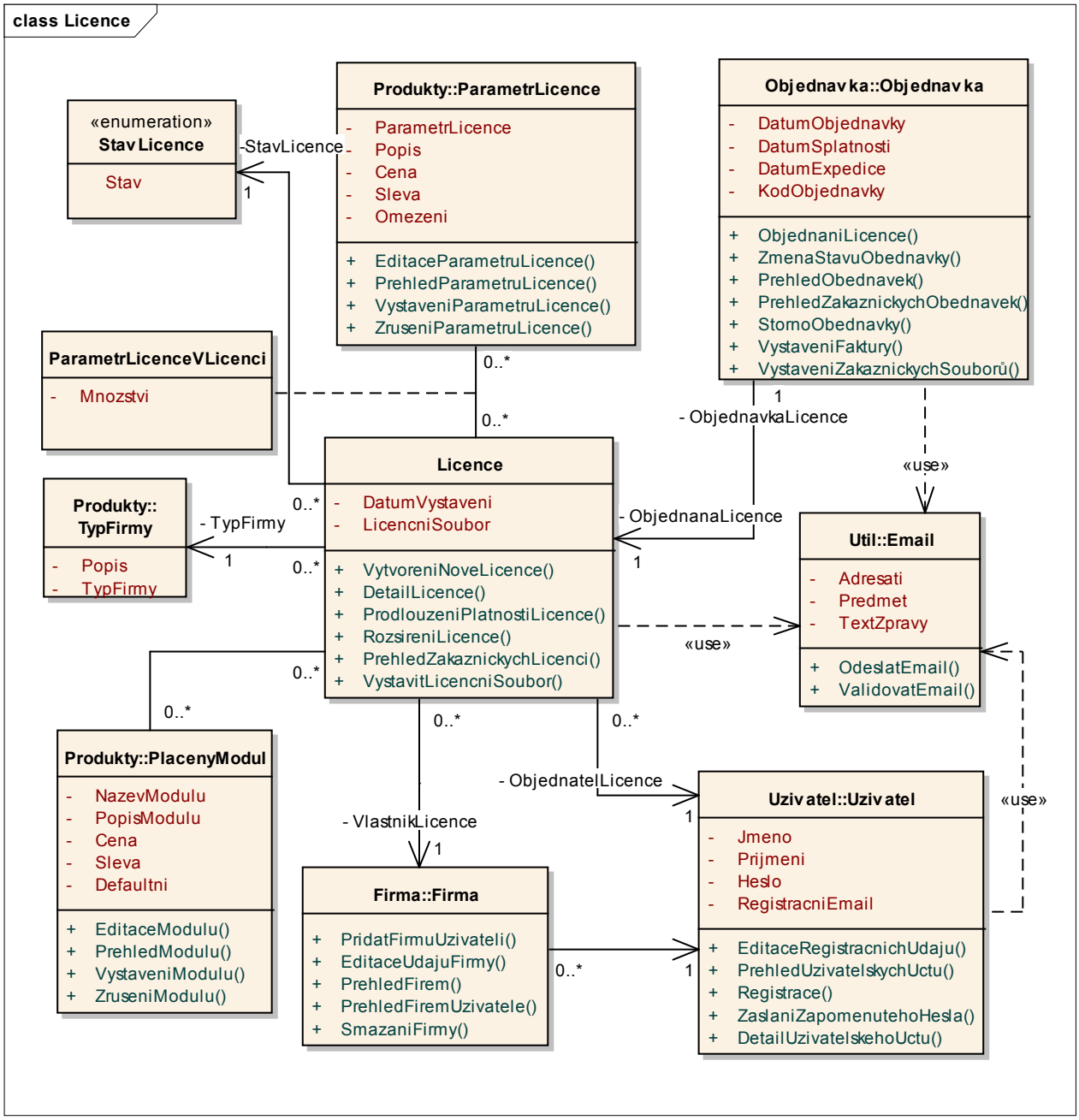
Databázi bude představovat aplikace třetí strany. Využije se k tomu Microsoft SQL server express 2005.



Obr.: 4.3.3 – Přehled použití aplikačních frameworků .NET v jednotlivých vrstvách

4.3.2 Model analytických tříd

Model analytických tříd je demonstrován na třídách z balíku licencí. Na obrázku 4.3.4 je vidět pouze hrubý analytický model tříd. Hlavní třídou v tomto balíku je třída *Licence*. Pro demonstraci některých důležitých vztahů jsou zde i třídy z jiných balíků (např. *Uzivatel, Objednavka, Firma* atd.).



Obr.: 4.3.4 – Analytické třídy – balík licence

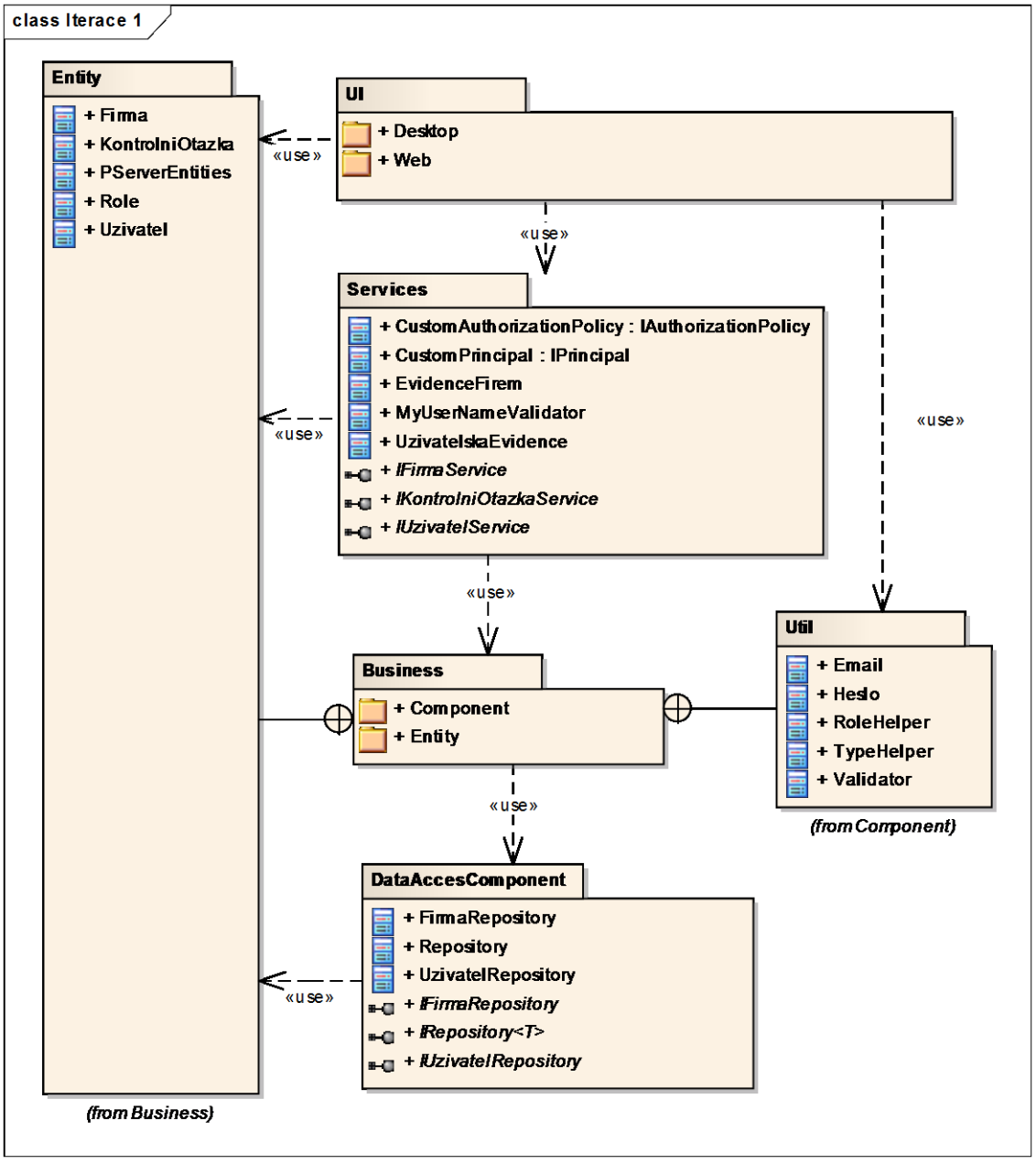
Z jednotlivých požadavků a případů užití byly převzaty do modelu základní funkce a některé atributy. Například v licenci bude vystupovat třída *StavLicence* v roli *StavLicence*. *StavLicence* určuje, v jakém stavu se licence nachází po svém vytvoření, objednání nebo stornování. Dále zde vystupuje v licenci *Uzivatel* v roli *ObjednatelLicence* a *Firma*. Licence bude vždy vlastněna na firmu = firma je v roli *VlastnikLicence*. Konkrétní *Uzivatel* je pak vzhledem k firmě v roli vlastníka firemního účtu. *Uzivatel* vystupuje vzhledem k licenci v roli *ObjednatelLicence* je to pro snadnější manipulaci s licencemi. Tato vazba je zde navíc.

V licenci dále vystupují třídy, které mají vliv na její podobu. Je to především 0-n (Píšu zde n ale modulů bude vždy konečný počet, zatím ale není jasné kolik přesně jich bude.) placených modulů, které si může uživatel do licence přidat. U třídy modulů je důležité zmínit parametr *Default*, který určuje, zdali je modul v licenci zahrnut defaultně (je potřeba pro chod aplikace už v základní verzi) nebo si jej uživatel bude moci přidat. Dále to jsou jednotlivé parametry licence určené třídou *ParametrLicence*. U parametrů licence zmíním pouze vlastnost asociace (*Mnozstvi*), každý parametr má totiž jednoznačně určeno své množství při vytváření uživatelem.

Třída *Email* z balíku *Util* je pouze používána ostatními třídami, proto je zavedena vazba se stereotypem <<*Use*>>. Znamená to, že instance třídy jsou vytvářeny pouze v rámci metod.

4.3.3 Návrh

V rámci návrhu demonstruju, jakým způsobem je řešen další postup při realizaci projektu a principy, které sem použil při řešení a následné implementaci. Všechny tyto principy jsou shodné pro většinu analytických tříd. Celý postup bude demonstrován na příkladu uživatelské evidence, jejíž návrh proběhl v rámci první iterace.

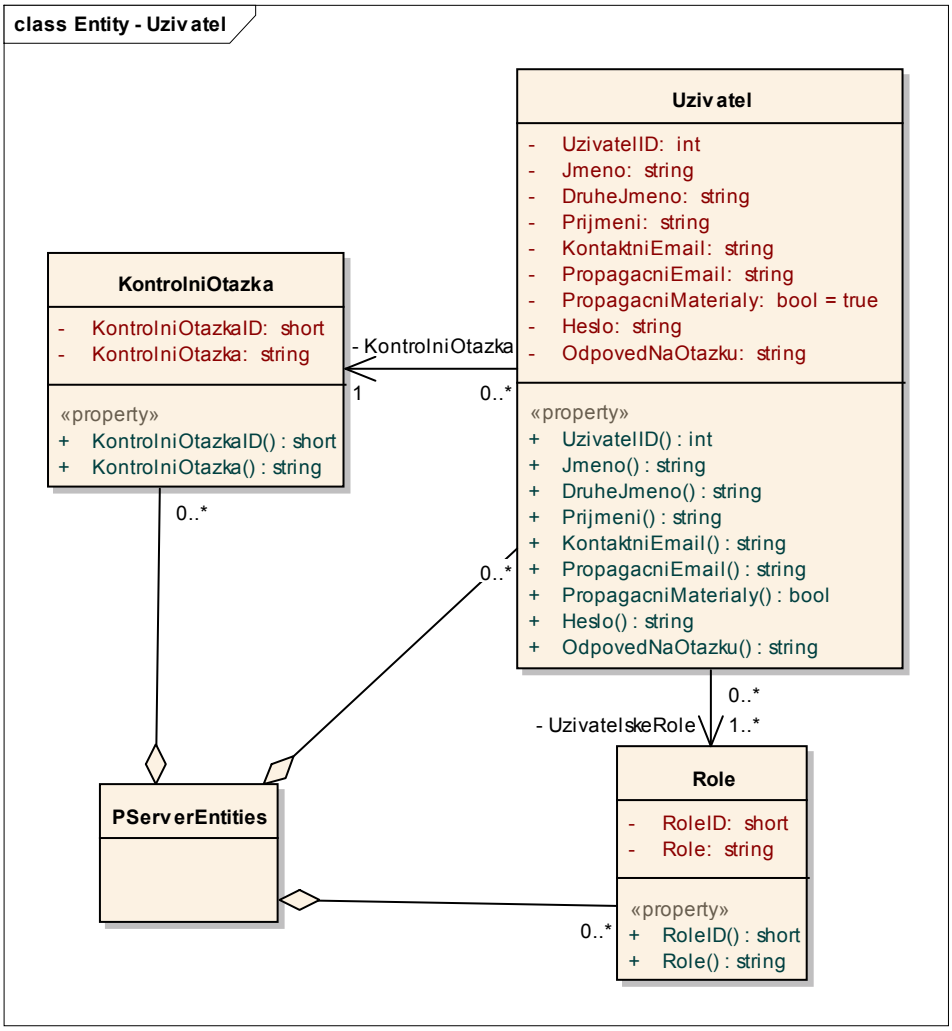


Obr 4.3.7 – Návrhové třídy v balicích – realizované v rámci první iterace

Na obrázku 4.3.7 je vidět, rozložení jednotlivých návrhových tříd v balících. Hlavní rozdíly oproti analytickým třídám je fakt, že díky použití Entity frameworku se mi každá prezistentní analytická třída rozpadla na dvě návrhové třídy. První zapouzdřuje atributy a tvoří tak entitu (v balíku entit) a druhá zapouzdřuje převážně operace a tvoří tak business třídu (v balíku *Business.Component*). Třídy z business vrstvy jsou pak určeny k provádění operací nad entitami.

Entity

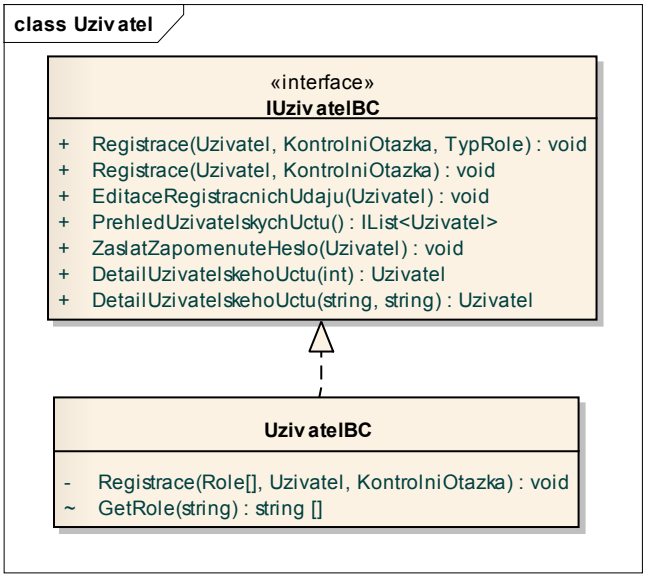
Na obrázku 4.3.8 je vidět třídní diagram pro entity balíku uživatel. Ty obsahují pouze atributy (properties) a některé základní operace. Všechny entity jsou také součástí *ObjectContextu*, který je v diagramu reprezentován třídou *PServerEntities*. Prostřednictvím tohoto kontextu potom dochází k dotazování do databáze a manipulaci s entitami.



Obr.: 4.3.8 – Diagram návrhových tříd – část diagramu entit z balíku uživatel

Business třídy

Obrázek 4.3.9 potom ukazuje třídu zapouzdřující business operace nad entitou uživatele. Třída *UzivatelBC* implementuje rozhraní *IUzivatelBC*, které předepisuje a třídě hlavní business operace (ty byly získány z analytických tříd + přidány některé speciální operace). Každá taková business třída implementuje nějaké veřejné rozhraní, které dává k použití ve vyšších vrstvách. Obecně všechny třídy, které nějakým způsobem poskytují funkcionalitu do vyšších vrstev, jsou v systému naprogramovány proti rozhraní. Tím je docíleno přísného oddělení jednotlivých vrstev.



Obr.: 4.3.9 – Třída a rozhraní uživatele z balíku Business.Component.Uzivatel

Vrstva pro přístup k datům

Pro přístup k datům a zjednodušení manipulace s entitami sem využil návrhový vzor repository. Vytvořil sem navíc jeho generickou verzi, takovým způsobem aby bylo možno použít základní CRUD + některé speciální operace pro všechny entity bez rozdílu typu.

Repository funguje tak, že generický interface *IRepository* předepisuje základní operace, které je možno provádět nad každou entitou. Tento interface potom dědí třída *Repository*, která tento interface reailzuje. *Repository* navíc implementuje základní operace nad object contextem – *PserverEntities*. Jedná se především o vytváření a ukončování spojení.

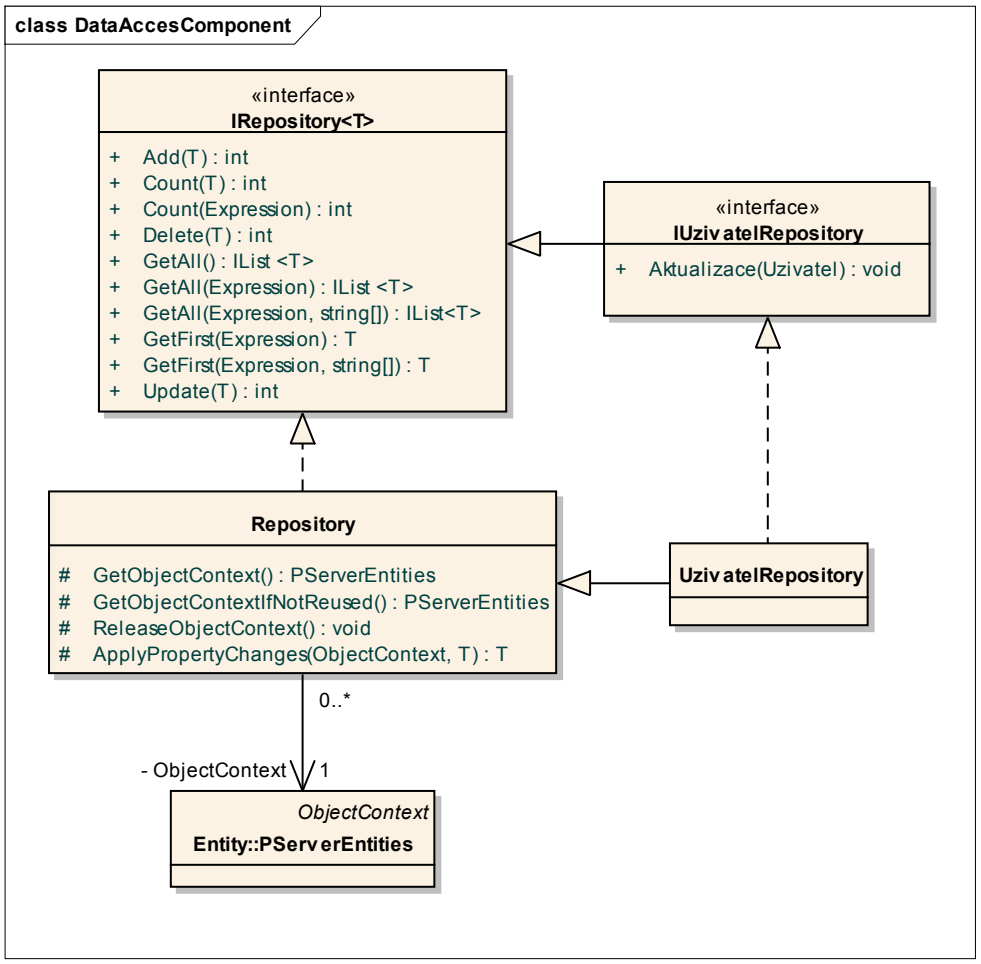
V programu poté stačí volat:

```
IRepository<Uzivatel> repository = new Repository<Uzivatel>();

Uzivatel t = repository.GetFirst(expression); //vrátí uživatele
```

V případě, že potřebuji nějaké konkrétní operace nad daným typem entity, můžu si v takovém případě implementovat konkrétní repository pro daný typ a operaci přidat do ní.

Na obrázku 4.3.10 je zobrazen třídní diagram pro konkrétní repository - *UzivatelRepository*, která implementuje jednu operaci *Aktualizace*. Konkrétní repository - *UzivatelRepository* se realizuje tak, že se nejprve *UzivatelRepository* podědí od *Repository* a navíc se realizuje rozhraní *IUzivatelRepository* (předepisuje operace které jsou navíc). Tím získávám už ne generickou ale typově zaměřenou konkrétní repository nad entitou *Uzivatel*.



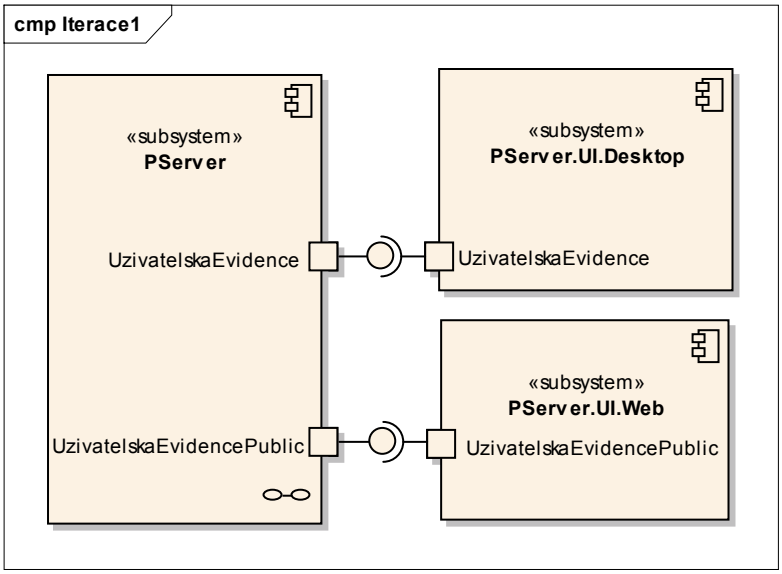
Obr.: 4.3.10 – Třídní diagram vrstvy pro přístup k datům – repositury uživatele

Služby

Vrstvu služeb, zde zmíním pouze stručně. Vrstva služeb je totiž pouze delegací konkrétních operací z business vrstvy ven z komponenty + aplikací příslušných bezpečnostních pravidel, které zajišťují třídy *CustomAuthorizationPolicy*, *CustomPrincipal* a *MyUserNameValidator*. Tyto třídy používá komponenta WCF automaticky na základě nastavení v konfiguračním souboru komponenty.

Uživatelské rozhraní

Vrstva uživatelského rozhraní obsahuje dvě komponenty (subsystémy) *UI.Web, UI.Desktop*, které implementují služby vystavené v servisní vrstvě.

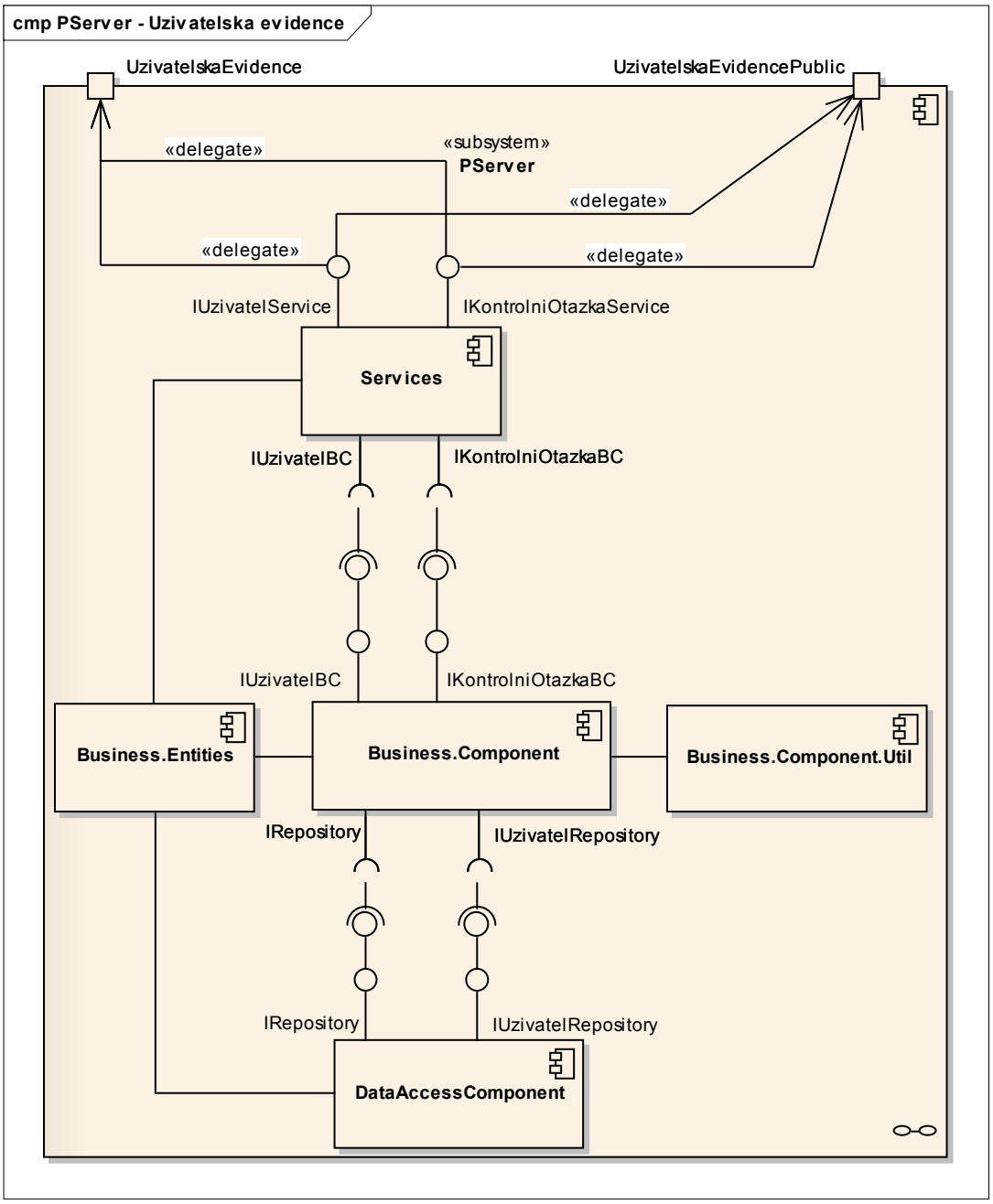


Obr.: 4.3.11 – Diagram komponent – rozdělení na subsystémy a jejich komunikace

Komponenty

Obrázek 4.2.11 ukazuje rozklad na subsystémy a jejich komunikaci. Subsystém *PServer* vystavuje ven z komponenty porty *UzivatelskaEvidence* a *UzivatelskaEvidencePublic*, které jsou implementovány jednotlivými subsystémy uživatelských rozhraní.

PServer je subsystém složený z dalších komponent jak ukazuje obrázek 4.3.12. Na tomto obrázku je vidět vnitřní uspořádání subsystému. Také je vidět, jak spolu komponenty uvnitř komunikují a jaké k tomu používají rozhraní. Nejzajímavější je delegace rozhraní *IUzivatelService* a *IKontrolniOtazkaService* z komponenty *Services* na porty *UzivatelskaEvidence* a *UzivatelskaEvidencePublic*, které jsou dostupné zvenčí a sdružují služby těchto rozhraní do jednoho místa (portu).



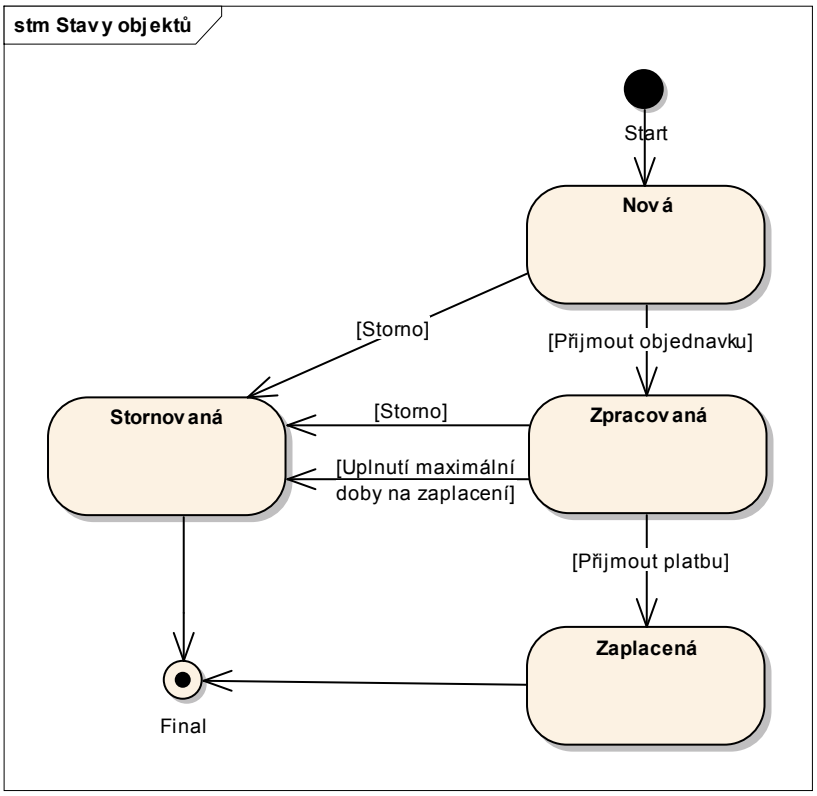
Obr. 4.3.12: – Komunikace komponent v subsystému PServer

4.3.4 Stavy některých objektů

V rámci návrhu by bylo dobré také zmínit a vysvětlit stavy některých objektů. Bude se jednat především o objekt Licence a Objednávky.

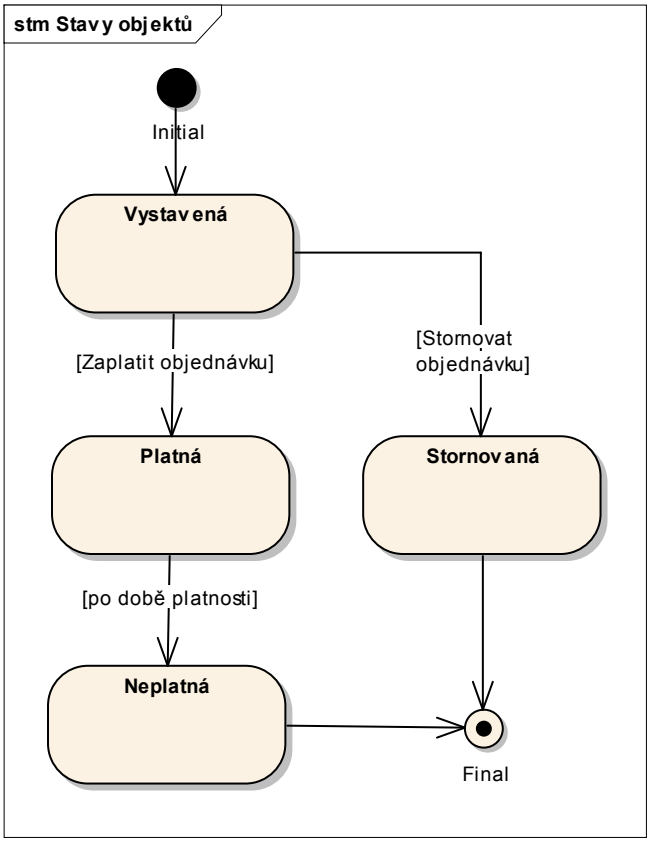
Objednávka bude mít 4 stavy. Při vytvoření objednávky uživatelem se bude nacházet ve stavu *nová*. Pokud uživatel objednávku stornuje, pak přechází do stavu *storno*. (Zde bude mít možnost uživatel zrušit objednávku v případě chybného objednání nebo pokud si to rozmyslí). Ze stavu

nová bude objednávka přecházet do stavu *zpracovaná*, pokud správce Pserveru objednávku zpracuje (např. vystaví a odešle fakturu atp.). Ze stavu *zpracovaná* zaplacením objednávky přechází objednávka do stavu *zaplacená* a tím jsme s objednávkou skončili. Ze stavu *zpracovaná* také může přejít do stavu *storno*, pokud uživatel do předem stanoveného termínu objednávku nezaplatí.



Obr. 4.3.13: Stavy objednávky

Stavy licence budou záviset do určité míry na stavu objednávky. Při založení licence (při vystavení objednávky) je licence ve stavu *vystavená*. Z tohoto stavu je ovlivněna právě stavem objednávky. Pokud dojde z jakéhokoliv důvodu ke změně stavu objednávky na *storno* pak se stornuje i vytvořená licence. V případě, že zákazník zaplatí svou vystavenou objednávku včas a objednávka tak přejde do stavu *zaplacená* pak i licence přechází ze stavu *vystavená* do stavu *platná*. Ze stavu *platná* pak po vypršení platnosti licence (záleží na jak dlouho zákazník zakoupí) přechází licence do stavu *neplatná*.



Obr. 4.3.14: Stavy licence

4.3.5 Datový model

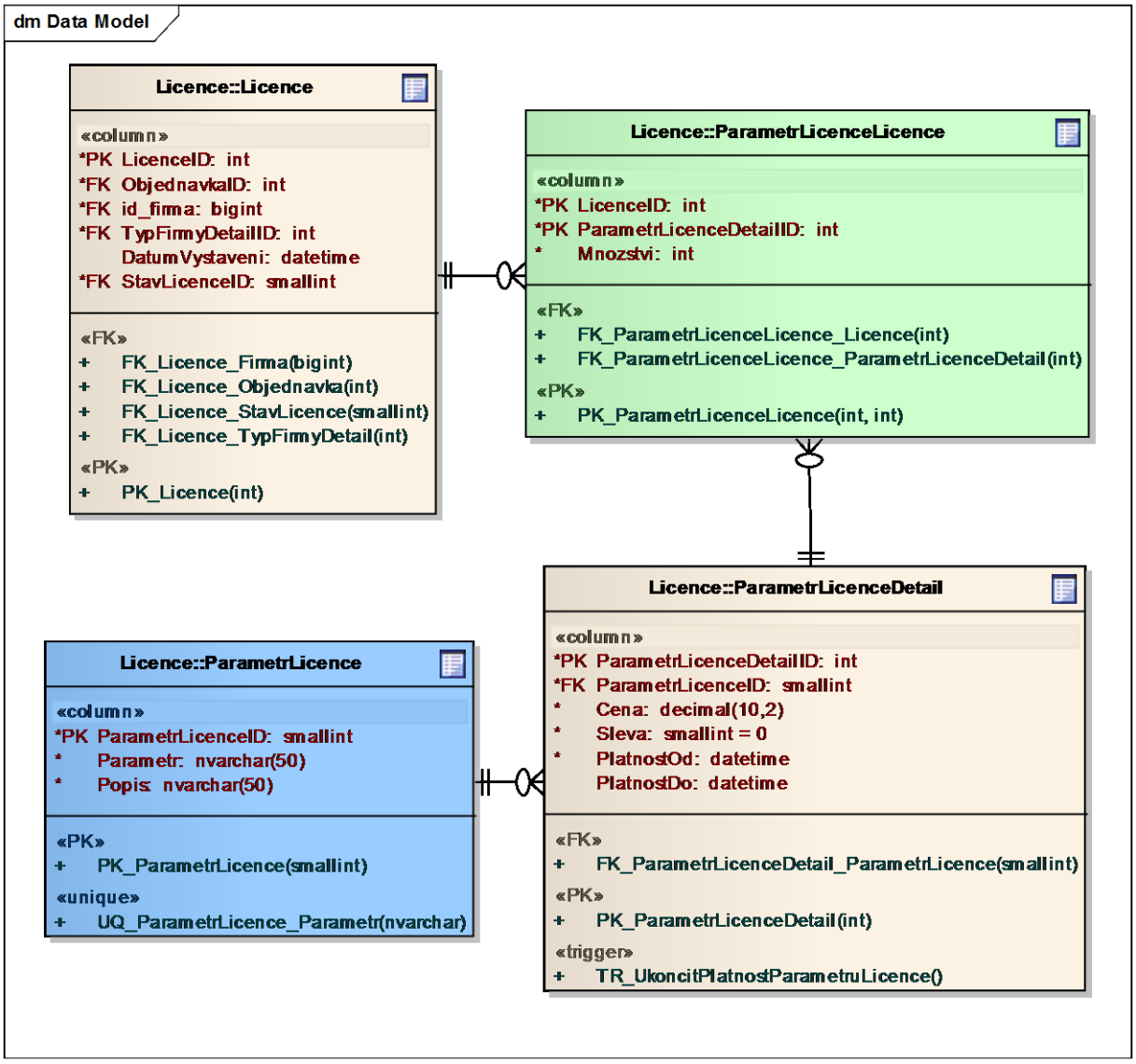
Datový model byl výchozí bod pro začátek implementace. Na základě relačního schématu sem vygeneroval z databáze pomocí entity frameworku konceptuální schéma (entity) se kterými sem dále pracoval. Datový model vesměs odpovídá třídnímu diagramu. Pouze některé konstrukce musely být rozloženy na jednodušší. Jednotlivé entity a jejich atributy sem taktéž převzal z třídních diagramů.

V datovém modelu sem taktéž sloučil některé atributy licencí pod jednu entitu a tou je *ParametrLicence*. Viz. obr. 4.3.15. Na obrázku je vidět část datového modelu. Tím sem si ušetřil některé tabulky navíc. K licenci se nicméně neváže přímo *ParametrLicence* (to je pouze číselník parametrů, které se dají do licence přidat) ale *ParametrLicenceDetail*. S tím souvisí následující odstavec o vedení historie dat.

U datového modelu je potřeba zmínit jeden požadavek, který bylo možno realizovat buď na úrovni business vrstvy nebo na úrovni databáze. Jedná se o archivaci některých dat, u kterých je uvedena cena. Tyto data, respektive jejich cena (ale mohou to být i další atributy) se mohou v čase měnit. Proto není možná editace takovýchto dat, ale při pokusu o změnu např. ceny se zavádí nový záznam do databáze s tím, že se starý zneplatní. Na obrázku 4.3.15 (zeleně jsou agregační tabulky, a modře číselníky) je vidět, tabulka *ParametrLicenceDetail*, která má za úkol vést historii k jednotlivým parametrům licence a ta je pak vázána na konkrétní licence. Tím je

dosaženo toho, že když se změní cena konkrétního parametru, cena u předchozích parametrů zůstane zachována. Mezi entity které potřebují zachovávat historii patří:

- všechny parametry licence (cena a omezení aplikace)
- typ firmy (cena a omezení aplikace)
- moduly přidávané do licencí (cena)



Obr.: 4.3.15 – Část datového modelu z balíku licence

Aby bylo možno historii evidovat zavedl sem k příslušným tabulkám trigger, který při vkládání záznamu najde a ukončí platnost předchozího záznamu a poté se vloží nový záznam. Tím sem vlastně volil realizaci na úrovni databáze. U všech tabulek (pro evidenci historie) navíc musí být zakázána možnost editace záznamu. To se ale musí dodržet na úrovni aplikace. Nicméně toto jsou již artefakty implementace.

4.4 Implementace - Použité technologie a nástroje

.NET 3.5

Aplikační framework nad kterým je vyvíjena i aplikace Agrokrom 6.0. Nejdříve bylo nutné posunout oba projekty z verze 2.0 na 3.5. To proběhlo hladce a tedy bylo možné použití WCF a Entity frameworku které sem chtěl použít pro jednotlivé komponenty a bez posunutí projektu na 3.5 by to nebylo možné.

ASP.NET, Windows forms

Pro webové rozhraní sem použil ASP.NET. Pro desktopovou aplikaci se nabízelo několik možností. Jednak to bylo windows forms, windows presentation foundation (WPF) nebo prvky rozhraní vyvinuté ve výzkumném ústavu pro Agrokrom 6.0, které jsou ovšem založeny na windows forms. Prvky GUI vyvinuté v Kroměříži sem zamítnul z toho důvodu, že ikdyž jsou udělány zdařile, stále se v nich vyskytují některé chyby, které znepříjemňují vývoj. Jako nejlepší volba by v současnosti bylo zvolit WPF. Výhoda je hlavně v možnosti ovlivňovat vzhled nezávisle na kódu a tedy měnit jednoduše výsledný vzhled aplikace. Další výhodou je využití vykreslování grafických prvků na grafické kartě. Nakonec sem se ale v rámci zachování kvality rozhodl pro klasické windows forms, se kterými mám na rozdíl od WPF více zkušeností. Pokud by ale v budoucnu bylo nutné změnit technologii nebyl by to problém. Prezentační vrstva je důsledně oddělena od zbytku aplikace a přechod by se týkal pouze rozhraní a ničeho jiného. Taktéž bych doporučil volit technologii WPF v případě budoucího transferu.

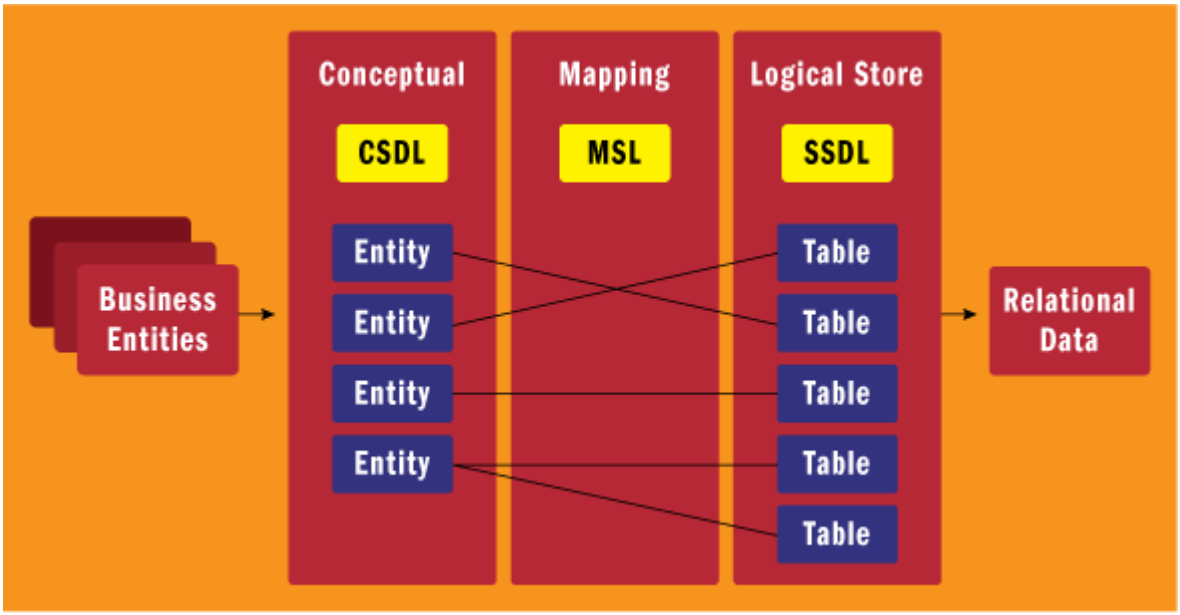
WCF – Windows communication foundation

Framework pro vystavení služeb. Komponenta, která zastřešuje jednotlivé komunikační technologie .NET frameworku a poskytuje k nim jednotný přístup. Tuto technologie jsem zvolil hlavně proto, že služby vystavené prostřednictvím WCF lze snadno konfigurovat prostřednictvím konfiguračního souboru který je založen na xml. Agrokrom nicméně používal starší typ .NET Remoting.

Entity framework - EF

V Agrokromu 6.0 bylo použito pro přístup k datům OOQL. OOQL je objektově relační maper který byl vyvinut v rámci prací na Agrokromu. Mě osobně se tento framework nezdá jako dobrá volba pro takovýto projekt. Mnohem lepší by bylo nasazení některého z existujících třeba i opensource frameworků (i v té době, kdy se začalo s vývojem Agrokromu už existovala portace javovského Hibernate na .NET nHibernate).

Pro zajištění nezávislosti dat na použité databázi sem zvolil technologii entity framework, která pochází přímo od Microsoft je její podpora a funkčnost je na úplně jiné úrovni než OOQL. Tento objektově relační maper zajistí mapování prezistentních entit z konceptuálního schématu na relační tabulky v databázi tak jak ukazuje obrázek 4.4.1. Výhodou je i možnost vygenerování entit přímo z databáze pomocí visual studia.



Obr.: 4.4.1 – Mapování v EF [17]

Enterpise Library 4

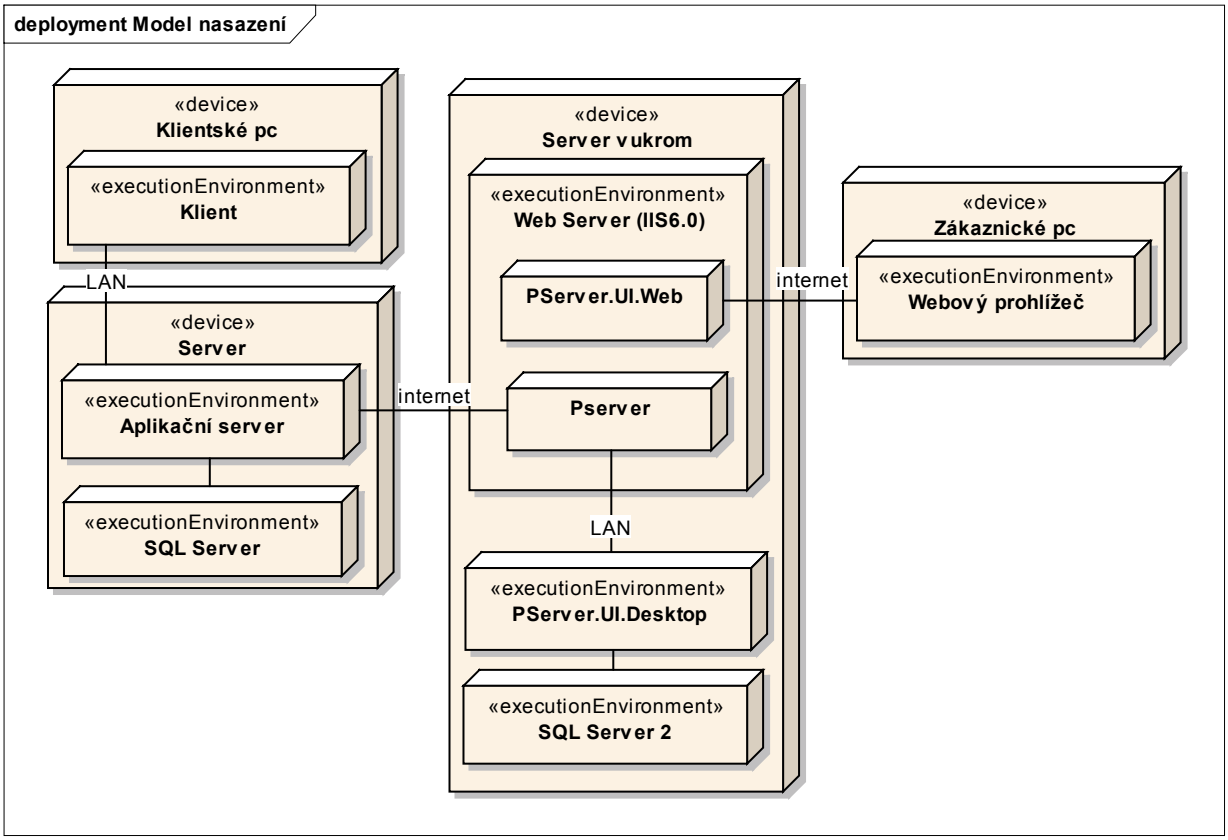
Aplikační knihovna implementující některé často používané scénáře při vývoji aplikací. Především logování, zachycování chyb, validace atd. Původně sem uvažoval využití této knihovny ve větším rozsahu ale nakonec sem se uchýlil k použití frameworku pouze pro logování.

Vývojové nástroje

- Pro programování aplikací sem použil Visual studio 2008. V případě, že budou použity express nástroje bude potřeba pro práci se službami a webovou aplikací použít Visual studio 2008 express a Visual web developer 2008 express. Nejlépe obě po aktualizaci na SP1.
- Pro modelování sem použil Enterprise Architect v licenci dostupné ve výzkumném ústavu v Kroměříži.
- Pro práci s databází sem použil SQL Server Management studio 2008.
- Jako verzovací systém sem použil Tortoise SVN, jež se používá i při vývoji Agrokromu.

4.5 Nasazení

Tato část popisuje hardwarovou konfiguraci a fyzické rozmístění jednotlivých programových komponent na HW zařízení. Na obrázku 4.5.1 je model nasazení aplikace.



Obr.: 4.5.1 – Model nasazení

Server vukrom – HW zařízení ve výzkumném ústavu v Kroměříži. Zde se budou spouštět všechny součásti PServeru. Musí být osazeno OS Windows server 2003 a frameworkem .NET 3.5. Jednotlivé součásti spolu budou komunikovat pomocí služeb.

- **PServer.UI.Desktop** – Administrační rozhraní pserveru . Samostatně spustitelná aplikace.
- **SQL Server 2** – Databázový server pro Pserver
- **Web Server (IIS)** - Webový server, na kterém bude hostována webová aplikace a aplikace služeb Pserveru.
    - **P-Server.UI.Web** - Webové rozhraní PServeru, určené zákazníkům a potencionálním zákazníkům. Komunikuje prostřednictvím lokální sítě s PServerem.
    - **PServer** – Samotná aplikace PServer běžící na Server vukrom jako služba

Zákaznické PC - Reprezentuje PC zákazníka kdekoliv ve světě, který bude přistupovat prostřednictvím webového prohlížeče k obchodní stránce PServeru.

- **Webový prohlížeč** - Jakýkoli moderní webový prohlížeč. Aplikace třetí strany, kterou si zákazník vybere a instaluje samostatně od aplikace Agrokrom a PServer.

Server - Servrové zařízení u zákazníka, na které se bude instalovat serverová část Agrokromu. Aplikační server a databázový server pro aplikace Agrokrom.

- **Aplikační server** – Aplikace pro správu Agrokromu a jeho klientů, která může komunikovat s PServerem prostřednictvím sítě internet.
- **SQL Server** – Databázový server pro Agrokrom

Klientské PC – HW zařízení reprezentující klienty Agrokromu

- **Klient** – klientská část Agrokromu instalována jednotlivým pracovníkům.

4.6 Popis iterací

Návrh aplikace byl rozdělen do několika částí (iterací). Každá z iterací bude zahrnovat všechny fáze vývoje a každá vyšší iterace pouze přidá dohodnutou funkcionalitu – inkrementální vývoj. Rozplánování jednotlivých fází sem převzal z [12] ale upravil sem si jej podle vlastních potřeb.

Iterace 1:

Úkoly:

- Seznámení se s architekturou a dokumentací k aplikaci Agrokrom a Aplikačnímu serveru. Ověření stavu těchto aplikací.
- Návrh systému licencí a způsobu distribucí pro jednotlivé části Agrokromu. Zvážit možnou technickou realizaci těchto problémů.
- Specifikace požadavků a případů užití.
- Základní návrh architektury aplikace P-Server.
- Získání shody mezi zadavatelem a realizátorem projektu na pochopení zadání.

Výstupy:

- Slovní zadání (Stakeholder request), dokument popisující aplikaci Agrokrom 6.0 a (její stav) a způsob jejího licenčního a distribučního systému, dokument vize, model požadavků, model případů užití, SAD - Software Architecture Document (pouze se základní koncepcí architektury).

Zhodnocení:

S architekturou Agrokromu jsem se částečně seznámil už při předchozí práci pro výzkumný ústav. Nicméně sem musel nastudovat její dokumentaci a práce některých členů, kteří se na vývoji podíleli. Taktéž jsem se musel seznámit s aplikačním serverem. Přitom jsem zjistil, že aplikační server není v takovém stavu, v jakém by měl být. Jádro aplikace bylo v rámci vývoje integrováno do klienta což pro účely vývoje Agrokromu postačovalo ale ne už pro vývoj Pserveru. Proto sem musel kontaktovat autora aplikace a dohodnout s ním rozdělení těchto dvou aplikací do stavu jaký je uveden v architektuře. Rozdělení aplikace proběhlo bez větších problémů.

V rámci rozdělení těchto aplikací sem taktéž volil technologie popsané v kapitole 4. Především šlo a WCF, které sem chtěl použít pro komunikaci aplikací. Proto bylo ale nutné posunout projekty na verzi .NET 3.5. To proběhlo opět bez větších potíží, pouze si to vyžádalo nastavení souborů projektů a některá další nastavení.

V rámci 3 schůzek, které se zadavatelem v tomto období proběhly, jsme se domluvili na výsledné podobě licencování a omezení aplikace a na některých základních funkcích, které budou v aplikaci potřeba.

Další práce už byly pouze v mé režii a šlo o vytvoření základní specifikace požadavků a nastínění architektury systému a vytvoření potřebných dokumentů, sloužících jako projektová dokumentace.

Iterace 2:

Úkoly:

- Vize projektu, je už neměnná.
- Kompletní analýza funkcí a případů užití
- Identifikace a upřesnění klíčových případů užití (identifikováno a analyzováno alespoň 80%)
- Na základě předchozího bodu dokončení návrhu architektury aplikace P-Server, která se už nebude nějak zásadně měnit.
- Implementace základních funkcí aplikace. Vytvoření prototypu P-Serveru, který bude zahrnovat i prototyp základního grafického rozhraní.

Výstupy:

- Model požadavků a model případů užití hotov minimálně z 80%, SAD, upřesnění modelu analýzy a návrhu, datový model, prototyp aplikace zaměřený hlavně na architekturu a integraci vybraných technologií.

Zhodnocení:

Tato fáze vývoje byla zaměřena z největší části na analýzu a návrh aplikace. Dokončil sem z větší části specifikaci požadavků a návrh aplikace. Taktéž jsem v této iteraci začal implementovat jádro systému. V uživatelských funkcích sem se zaměřil především na evidenci firem a uživatelů. Šlo mi o to, abych obsáhl všechny vrstvy aplikace až po uživatelské rozhraní a vytvořil si tak lepší představu o tom jakým způsobem budu v dalších iteracích pokračovat v implementaci uživatelských funkcí. A také o to abych mohl demonstrovat alespoň základní funkční prototyp aplikace.

V této fázi jsem narazil na několik problémů, které se týkaly použití mnou vybraných technologií. V první řadě šlo o zabezpečení komunikace mezi aplikacemi. K tomu bylo nejprve nutné prostudovat manuál o bezpečnosti k WCF [13]. V rámci WCF sem volil message security (zabezpečení na úrovni zpráv). To si ale vyžádalo zabezpečení certifikátem, který sem nejprve musel získat a následně i správně aplikovat. Pro testovací účely sem využil aplikaci makecert.exe od Microsoftu prostřednictvím které sem si vygeneroval vlastní testovací sadu certifikátů. No pro reálné použití by bylo vhodné použít certifikát generovaný nějakou certifikační autoritou.

Na další problémy sem narazil při vývoji na různých strojích a při přenášení projektu mezi různými verzemi operačního systému Windows. I když sem s cílovou platformou (Windows server 2003 a IIS 6) počítal dopředu a snažil jsem se podchytit všechny její omezení už na

začátku projektu, nemohl jsem na této platformě vyvíjet. Primárně sem proto vyvíjel na Windows 7 a IIS 7 což pro mě bylo nejdostupnější řešení. Nicméně sem se musel uchýlit i k použití Windows XP a IIS 5 a k vývoji použít express nástroje od Microsoftu.

Různé verze přinesly potíže v podobě chyb v aplikaci, kdy přestávaly fungovat především funkce starající se o zabezpečení komunikace, certifikáty a transakční zpracování atd. Tyto funkce, jak sem se měl možnost přesvědčit, do značné míry závisely na nastavení OS a IIS. Jenže v každé z verzí operačního systému bylo potřeba k nastavení stejné funkcionality různého přístupu a někdy i různých technologických prostředků. To značně zpomalovalo implementaci protože, chyby v aplikaci způsobené těmito problémy se těžko odhalovaly a značně zdržovaly implementaci funkcí, které sem v této iteraci naplánoval.

Iterace 3:

Úkoly:

- Implementace většiny funkcí

Výstupy:

- Kompletní zpracování všech požadavků (analýza a návrh) zahrnující opravy z předchozí iterace.
- První funkční verze systému
- Testování aplikace

Zhodnocení:

Tato iterace byla zaměřena převážně na implementaci většiny zbylých funkcí a jejich programátorské testování. Analýza a návrh aplikace byly v této chvíli hotovy ale i přesto docházelo k menším úpravám a změnám ve všech fázích vývoje. Převážně se jednalo o opravy mnou způsobených chyb v návrhu atd. Většinu problémů sem vyřešil během předchozích dvou iterací.

Iterace 4:

Úkoly

- Dokončení implementace zbylých funkcí
- Integrace funkcí do aplikačního serveru
- Oprava funkcí podle požadavků zadavatele
- Testování a nasazení aplikace
- Dokumentace

Výstupy

- Funkční verze systému + dokumentace

Zhodnocení

V rámci této iterace bylo na řadě implementovat zbylé funkce. Provést opravy na základě připomínek zadavatele a integrovat funkce do aplikačního serveru Agrokromu. Po ukončení implementačních částí provést testy a nasazení aplikace.

Tím, že v předchozích iteracích vzniklo zdržení, sem nebyl schopen v rámci projektu realizovat tuto iteraci v plném rozsahu. Pro co nejlepší kvalitu aplikace, měla proběhnout další schůzka se zadavatelem, kde by se dohodly opravy a připomínky k hotovým funkcím z třetí iterace. Schůzka ale díky časovému skluzu neproběhla a byl sem nucen plynule přejít k realizaci dalších bodů v zadání a specifikaci tak abych splnil rozsah práce. Jednalo se především o dokončení dokumentace projektu a provedení základních testů a nasazení aplikace.

5 Závěr

Cílem této diplomové práce bylo navrhnutí a technická realizace aktualizačního serveru. Jeho primární funkcí jak napovídá název, bylo poskytování aktualizací pro již existující informační systém Agrokrom 6.0. V průběhu práce, hlavně během specifikace požadavků, se rozsah práce změnil. Na základě několika schůzek se zadavatelem bylo domluveno, že některé dosavadní činnosti, které se týkají distribuce Agrokromu by mohly být plně nebo alespoň částečně automatizovány prostřednictvím tohoto serveru. Jednalo se především o funkce pro správu licencí zákazníků, objednání těchto licencí a následnou distribuci k zákazníkům tak, jak je popisuji v tomto textu. Přidání těchto funkcí trochu mění pohled na aktualizační server a činí tento název mírně zavádějící. Lepší pojmenování by mohlo být, server podpory zákazníků. Takový název by snad lépe vystihl charakter aplikace.

V rámci práce sem se seznámil s aplikacemi Agrokromu 6.0 a jejich dosavadním stavem. Na základě dostupné dokumentace jsem navrhnul a realizoval programové vybavení dle specifikace požadavků. Vznikly tak vlastně 3 menší aplikace. Jedna aplikace poskytující služby a dvě aplikace uživatelských rozhraní (webové a desktopové administrační). Při realizaci ve 2 iteraci sem narazil na několik problémů, které bylo nutné vyřešit a poměrně mě v realizaci zdržely. Díky zdržení sem se musel odklonit od některých cílů důležitých pro zadavatele a realizovat projekt tak, aby vyhověl původnímu zadání diplomové práce. Tím vzniklo, pár dalších požadavků, které bude nutno pro bezproblémové nasazení aplikace potřeba realizovat. Doporučuji tedy provést ještě další iteraci na doplnění a vyladění některých funkcí do finálního stavu tak, aby byla aplikace schopná bezproblémového nasazení a provozu.

6 Literatura

[1] ŠTĚDROŇ, Bohumír. *Ochrana a licencování počítačového program*. Praha : Wolters Kluwer ČR, 2010. 220 s. ISBN 978-80-7357-555-7

[2] Kategorie svobodného a nesvobodného software - http://www.gnu.org/philosophy/categories.cs.html#PublicDomainSoftware

[3] Softwarová licence - http://cs.wikipedia.org/wiki/Softwarov%C3%A1_licence

[4] Copyleft - http://cs.wikipedia.org/wiki/Copyleft

[5] Google chrome EULA - http://www.google.com/chrome/intl/cs/eula_text.html

[6] Freeware - http://cs.wikipedia.org/wiki/Freeware

[7] GNU GPL - http://www.gnu.cz/article/32/

[8] Public domain -http://cs.wikipedia.org/wiki/Voln%C3%A9_d%C3%ADlo

[9] GNU LGPL - http://www.gnu.cz/article/34/

[10] GNU FDL - http://www.gnu.cz/article/36/

[11] PECINOVSKÝ, Rudolf. *Návrhové vzory*. Brno : Cpress, 2007. 527 s. ISBN 978-80-251-1582-4

[12] ARLOW, Jim; NEUSTADT, Ila. *UML2 a unifikovaný proces vývoje aplikací.* Brno : Cpress, 2008. 567 s. ISBN 978-80-251-1503-9.

[13] WCF security Guidance - http://wcfsecurity.codeplex.com/

[14] prof. Ing. Ivo Vondrák, CSc. *Úvod do softwarového inženýrství*.Vysoká škola báňská - Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, 2002.

[15] Iterative development - http://www.thefullwiki.org/Iterative_development

[16] Dokumentace Agrokromu[3]

[17] ADO.NET Entity Framework Overview - http://msdn.microsoft.com/en-us/magazine/cc163399.aspx#edupdate

[18] Projektová dokumentace k PServeru[4]

[3,4] Kompletní dokumentaci, je možno po domluvě shlédnout u zadavatele.